TIETO- JA SÄHKÖTEKNIIKAN TIEDEKUNTA

Jukka Pajukangas
Jussi Kurikka
Lasse Hyyryläinen

SUORAKULMAISEN PINNAN TUNNISTAMINEN LISÄTYN TODELLISUUDEN SOVELLUKSESSA

Kandidaatintyö
Tietotekniikan tutkinto-ohjelma
Heinäkuu 2017

Pajukangas J, Kurikka J, Hyyryläinen L. (2017) Suorakulmaisen pinnan tunnistaminen lisätyn todellisuuden sovelluksessa. Oulun yliopisto, tietotekniikan tutkinto-ohjelma. Kandidaatintyö, 49 s.

TIIVISTELMÄ

Lisättyä todellisuutta (Augmented Reality, AR) on tutkittu jo jonkin aikaa monenlaisten ammattialojen käyttöön, ja älylaitteiden yleistyessä viihteellinen lisätty todellisuus on myös yleistynyt. Lisätyn todellisuuden sovelluksia on laajasti kehitetty sotilas-, teollisuus-, opetus- ja viihdekäyttöön. Jotta lisätyn todellisuuden sovellukset voisivat toimia tehokkaasti, on sovellusten kyettävä ympäristön havainnoinnin lisäksi muodostamaan havainnoistaan sovelluksen kannalta tarkoituksenmukaisia tulkintoja. Suurimpia haasteita lisätyn todellisuuden sovelluksien kehityksessä on siis sovelluksen saamien havaintojen nopea ja luotettava prosessointi.

Tutkimuksen yhteydessä kehitettiin peli, jossa virtuaalinen pallo lisätään pelilaitteen näytölle, ja pelaajan tehtävänä on liikuttaa kättään kameran edessä ja koskea palloa saadakseen pisteitä. Tutkimuksessa hyödynnettiin minitietokone Raspberry Pi2:ta, johon liitettiin kosketusnäyttö sekä kamera. Tutkimuksessa keskityttiin suorakulmaisen pelialueen tunnistamiseen erilaisissa olosuhteissa, kuten huono valaistus tai kun osan pelialueesta peittää jokin tunnistamista häiritsevä este. Pelialueen tunnistamisprosessissa voitiin valita kahdesta erilaisesta segmentointitavasta, Cannyn reunantunnistusalgoritmia hyödyntävästä segmentoinnista tai Otsun metodista. Ohjelman käyttämään suodatukseen pystyi myös valitsemaan monta erilaista suodatustapaa, kuten mediaanisuodatuksen ja Gaussin suodatuksen.

Testeistä saatiin monia suuntaa antavia tuloksia. Cannyn reunantunnistusalgoritmin hyödyntäminen auttoi tunnistamaan pelialueen paremmin, kun osa alueesta oli peitetty ja kun pelialue oli kallistunut. Hyvissä olosuhteissa molemmat tunnistusmenetelmät toimivat yhtä hyvin. Otsun metodi auttoi paremmin tunnistuksessa huonossa valaistuksessa sekä oli hieman nopeampi Cannya hyödyntäneeseen metodiin verrattuna.

Avainsanat: kuvan prosessointi, kuvan reaaliaikainen prosessointi, suorakulmaisen alueen etsintä, Canny reunantunnistusalgoritmi, Otsun metodi

Pajukangas J, Kurikka J, Hyyryläinen L. (2017) Detection of rectangular surface in augmented reality application. University of Oulu, Degree Programme in Computer Science and Engineering. Bachelor's Thesis, 49 p.

ABSTRACT

Augmented Reality (AR) has been researched for years to be used among several professional areas and since smart devices are becoming more and more common, entertainment use in augmented reality has also become more common. Augmented reality has been applied to military, education, manufacturing and entertainment use. So that augmented reality software can work efficiently, the software needs to be able to detect the environment and make appropriate interpretations. One of the biggest challenges in augmented reality software development is to quickly and reliably process observations.

With the research, a game was developed where a virtual ball is added to the devices screen. The players task is to touch the ball while their hand is in front of the camera to score points. The research utilized a minicomputer, Raspberry Pi2, with a touchscreen and a camera attached. The research focused on finding a contour rectangular area in various kinds of circumstances, for example dim lighting or when an object is preventing accurate detection. The detection process for identifying the contour area has two different choices of segmentation methods, a segmentation method that utilizes Canny edge detection algorithm or Otsu thresholding. For filtering, one could choose from multiple filtering methods, for example median filtering or Gauss filtering.

The tests gave several approximate results. Canny edge detection utilizing segmentation assisted the detection process better when some of the area was obstructed and when the area was tilted. Both, Otsu segmentation and the Canny edge detection utilizing segmentation assisted the algorithm equally well in good circumstances. Otsu method assisted the detection process better in dim lighting and was slightly faster in calculations.

Key words: finding rectangular area, image processing, real time image processing, Canny edge detection algorithm, Otsu method

SISÄLLYSLUETTELO

ALKULAUSE

Haluamme kiittää Oulun yliopiston tieto- ja sähkötekniikan tiedekuntaa tärkeistä neuvoista ja joustavuudesta. Erityisesti haluamme kiittää professori Juha Röningiä sekä Teemu Tokolaa joustavuudesta ja avusta. Haluamme myös kiittää läheisiämme sekä ystäviämme tärkeästä tuesta.

Oulu, heinäkuu 6. 2017

Jukka Pajukangas
Jussi Kurikka
Lasse Hyyryläinen

LYHENTEIDEN JA MERKKIEN SELITYKSET

AR	Augmented Reality
HMD	Head-Mounted Display
HUD	Heads-Up Display
USB	Universal Serial Bus
WLAN	Wireless Local Area Network
RAM	Random Access Memory
DSI	Display Serial Interface
CSI	Camera Serial Interface
GPIO	General Purpose Input/Output
FPS	Frames Per Second
IIR	Infinite Impulse Response
OTK	Oikein tunnistetut kulmat
VTK	Väärin tunnistetut kulmat
VTK_VKA	Väärin tunnistettujen kulmien virheen keskiarvo
SA_KA	Suoritusajan keskiarvo

1. JOHDANTO

Lisätty todellisuus tarkoittaa sananmukaisesti todellisuutta, johon on lisätty jotakin lisäinformaatiota. Tämä lisäinformaatio voi olla esiintyä ihmiselle missä tahansa aistein havaittavissa olevana asiana, kuten esimerkiksi kuvina, ääninä tai jopa haju- ja tuntoaistimuksina. Ideat, jotka syntyvät kokonaan uuden kerroksen lisäämisestä havaittuun todellisuuteen ovat erittäin kiintoisa alue tutkimukselle ja tekniikan kehitykselle. Lisätyn todellisuuden mahdollisuuksien rajat ovat vielä tutkimattomia. Lähitulevaisuudessa saattaisi internetin sisältämän tiedon yhdistäminen todellisuuteen lisätyn todellisuuden kautta nousta ajankohtaiseksi asiaksi. Vielä pidemmälle tulevaisuuteen katsottaessa lisätty todellisuus voisi jopa sulautua erottamattomaksi osaksi ihmisten havainnointikykyä esimerkiksi kehoon asennettavien implanttien avulla.

Nykyhetken tilannetta tarkasteltaessa on lisätyn todellisuuden sovelluksia esiintynyt monipuolisesti eri käytännön aloilla. Lisätyn todellisuuden soveltaminen on synnyttänyt hyödyllisiä sovelluksia teollisuuteen [1], rakentamiseen [2], opetukseen [3] ja sotilaskäyttöön [4]. Spatiaalisia lisätyn todellisuuden sovelluksia [5] on hyödynnetty esimerkiksi museoissa (kuva 1). Nykyisten sovellusten kehitystä on auttanut käytettävien laitteiden hintojen laskeminen, suorituskyvyn paraneminen ja koon pieneneminen. Edellä esitetyt seikat ovat johtaneet lisätyn todellisuuden sovelluksien toteuttamisen helpottumiseen ja siten tarjonnut mahdollisuuden uusille ideoille toteutua.

Jotta lisätyn todellisuuden sovellus pystyisi lisäämään lisäinformaatiota todellisuuteen halutulla tavalla, on sovelluksen pystyttävä jollakin keinoin havainnoimaan ympäristöään. Mahdollisia havainnoinnin tapoja ovat esimerkiksi kameralla kuvaaminen, mikrofonilla kuuntelu tai sijainnin tai asennon tunnistaminen. Käytännössä pelkällä ympäristön havainnoinnilla ei saada aikaan hyödyllisiä sovelluksia, vaan sovelluksen on osattava myös tulkita havaintojaan tuottaakseen hyödyllisiä tietoja lisättäväksi todellisuuteen, eli sovelluksen käyttäjän näkyviin. Tämä tulkitseminen voisi tarkoittaa esimerkiksi ihmisen tunnetilan päättelyä hänen puheestaan tai liikkuvan kappaleen tunnistamista tai nopeuden laskemista videokuvan perusteella.

Niissä lisätyn todellisuuden sovelluksissa, joissa hyödynnetään kuvallista havainnointia ympäristöstä, on usein tarkoituksena tunnistaa kuvasta jokin sovelluksen tarkoituksen kannalta kiinnostava kohde. Tämä vaatii usein monenlaisten kuvan prosessoinnin tekniikoiden ja menetelmien soveltamista kulloisenkin tehtävän mukaisesti. Uusien tekniikoiden kehittäminen ja testaaminen tällä alalla on erityisen tärkeää, koska ei ole yhtä universaalia ratkaisua jokaiseen olemassa olevaan ongelmatilanteeseen. Uusi tieto kohteiden tunnistamisesta kuvasta tulee olemaan hyödyksi lisätyn todellisuuden alan lisäksi myös koko tietokonenäön tutkimuksen suhteen.

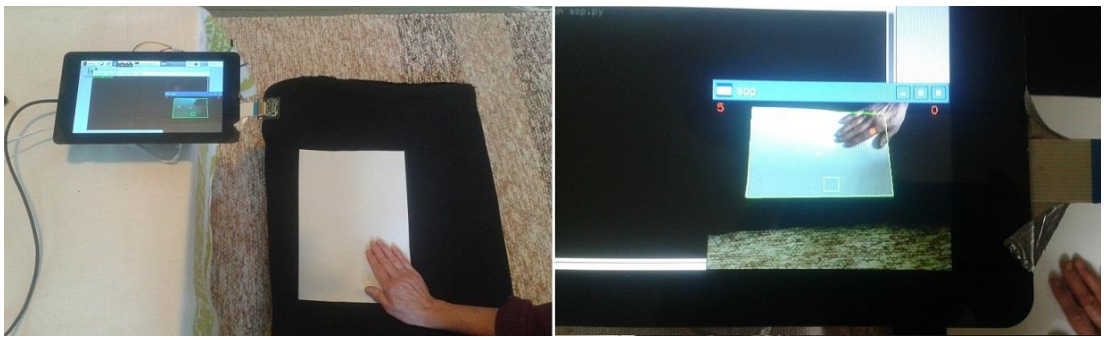
Kohteiden havaitsemista kuvasta tarvitsevat lisätyn todellisuuden sovellukset voivat asettaa tiukkoja vaatimuksia tämän ominaisuuden toiminnalle. Kohteiden kuvasta tunnistamisen varmuus on erittäin tärkeää, koska sovelluksen käytön mielekkyys on usein suoraan verrannollinen kohteen löytämisen varmuuteen. Toinen tärkeä vaatimus on tehtävään kuluva aika. Tehtävän kuluttama suoritusaika on yleensä oltava mahdollisimman alhainen, koska lisätty todellisuus on luonteeltaan reaaliaikaista. Edellä mainitun kaltaisissa sovelluksissa on kriittisen tärkeää saada

nämä kaksi vaatimusta täyttymään, jotta sovellukset olisivat mahdollisimman hyödyllisiä sekä käyttömukavia nyt ja tulevaisuudessa.

Tämä työ esittelee lisätyn todellisuuden pelisovelluksen (kuva 2). Sovellus hyödyntää minitietokonetta ja siihen liitettyä kameraa sekä kosketusnäyttöä. Sovellus lisää kameran havaitsemaan pelialueeseen virtuaalisen pallon, jota pelaajan on kosketettava kädellään saadakseen pisteitä. Sovellus hyödyntää pelialueen tunnistavassa algoritmissa kahta erilaista segmentointitapaa käyttäjän valinnan mukaan.

Esiteltävän sovelluksen keskeisenä ominaisuutena on pyrkiä tunnistamaan luotettavasti ja tehokkaasti videokuvasta pelialueena toimiva A4-paperiarkki. Tämä ominaisuus on olennaisin sekä esiteltävän sovelluksen kannalta, sekä yleisesti lisätyn todellisuuden kappaleiden tunnistamisen kannalta. Sovelluksen pelialueen tunnistavaa algoritmia testattiin erilaisissa vaihtelevissa olosuhteissa, kuten tilanteissa, joissa pelialue oli osittain peitetty tai valaistusolosuhteet olivat huonot tai kameran sijainti pelialueeseen nähden oli epätavallinen. Tuloksiin kerättiin tiedot algoritmin suorituskyvystä ja suoritusajoista eri olosuhteissa ja eri algoritmin alkuasetuksia käyttäen. Suoritetut testit ja tulokset esitellään luvussa 5 ja luvussa 6 analysoidaan näiden testien tuloksia ja pohditaan luodun sovelluksen jatkokehityksen mahdollisuuksia.

Kuva 1. Museon spatiaalinen näyttö.



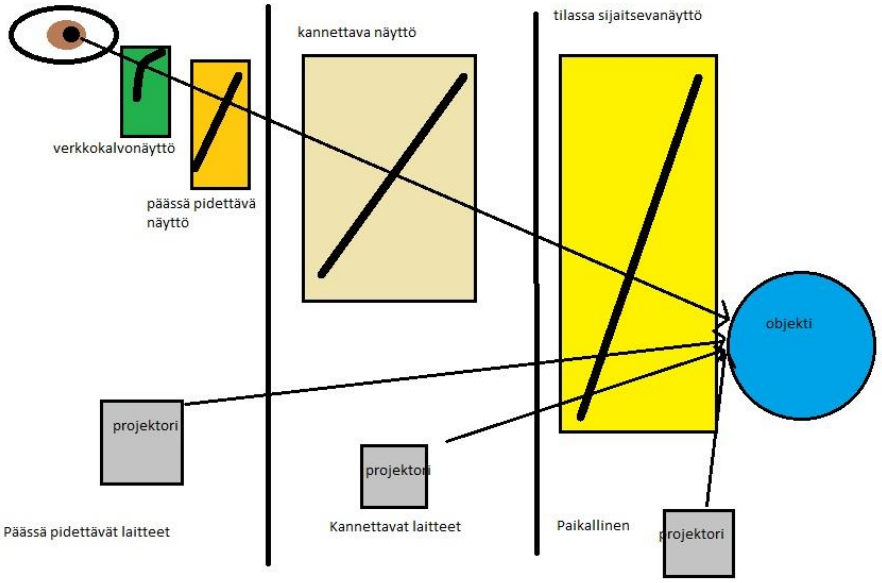
Kuva 2. Pelaaja käyttää tässä työssä tehtyä pelisovellusta.

2. LISÄTYN TODELLISUUDEN KÄYTÄNNÖN SOVELLUKSIA

Lisätyn todellisuuden on ennustettu mullistavan tavan, jolla ihminen voi aistia maailmaa. Lisättyä todellisuutta on sovellettu jo jonkin aikaa erilaisiin teollisuuden, viihteen sekä oppimisen tarkoituksiin, mutta laitteiden tehojen noustessa ja älylaitteiden yleistyessä myös lisättyä todellisuutta on helpompi tuoda kaikille tarjolle. Voitaneen myös ennustaa, että lisätty todellisuus tulee ottamaan tulevaisuudessakin suuremman ja suuremman roolin ihmisten elämässä auttaen kaikenlaisissa tehtävissä. Lisätyn todellisuuden laitteita on jo monessa paikassa monessa eri muodossa, kuten päässä pidettävinä laseina, joilla voidaan tunnistaa panssaroidun ajoneuvon korjattavat osat, tauluina, jotka tuovat ympäristöön jo kauan kuolleina olleita lajeja sekä älypuhelimina, jotka pystyvät näyttämään miltä tatuointi näyttäisi käyttäjän iholla.

2.1. Lisätyn todellisuuden laitteet

Lisätyn todellisuuden laitteet voidaan jakaa kolmeen suureen ryhmään; päässäpidettäviin laitteisiin, kannettaviin laitteisiin ja tilassa sijaitseviin laitteisiin, laitteenkannettavuuden ja liikuteltavuuden mukaan (kuva 3).



Kuva 3. Päässä pidettävien, kannettavien ja paikallisten näyttöjen esittely

2.1.1. Päässä pidettävät laitteet

Päässä pidettäviin laitteisiin kuuluvat verkkokalvonäytöt [6][7] ja päässä pidettävät näytöt eli kypäränäytöt [8][6]. MicroVision on kehittänyt lasien päälle asetettavaa Nomad retinal scanning display-verkkokalvonäyttöä [6]. Laite pystyy näyttämään vain yhden värin yhdelle silmälle [6]. Syy verkkokalvonäyttöjen vähäisyyteen ja teknologisiin puutteisiin, kuten yksivärisyys, ovat laitteiden suuret koot sekä

puuttuva teknologia luoda tarpeeksi pieniä ledejä [9]. Verkkokalvonäytöissä käytettävien lasereiden turvallisuutta on myös tutkittu, ja verkkokalvonäytöt on todettu turvalliseksi sopivilla tehomäärillä [7]. Verkkokalvonäyttöjä on tutkittu myös lentäjien tarvitseman informaation esittämiseen [10]. Kypäränäyttöjä, eli Head-Mounted Display:ta (HMD) tai Head-Up Display (HUD), käytetään esimerkiksi lentokoneissa kypärään liitettynä [11] sekä huoltotöissä [12][13]. HMD-tekniikka on tullut suuremman yleisön tietoon Googlen Google Glass:n myötä. Lisäksi Microsoft on kehittämässä omia HMD-lasejaan, joiden nimi on HoloLens [14].

2.1.2. Kannettavat laitteet

Kannettaviin laitteisiin kuuluvat kaikki kannettavat laitteet, kuten älypuhelimet, tabletit ja kannettavat projektorit [6]. Suurelle yleisölle kannettavat laitteet ovat paras tapa esitellä lisättyä todellisuutta laitteiden yleisyyden takia. Tämän lisäksi laitteiden prosessointitehot ovat moninkertaistuneet viime vuosikymmenien aikana [15]. Nämä syyt tekevät kannettavista laitteista erinomaisia lisätyn todellisuuden suuren yleisön laitteita. Tämän takia kannettavien laitteiden käytettävyyttä tutkitaan paljon [15]. Kannettaville laitteille on valmistettu jo paljon sovelluksia, kuten pelejä [16], navigoinnin avustajia [17] sekä mainossovelluksia [17].

2.1.3. Tilassa sijaitsevat näytöt

Tilassa sijaitsevat näytöt tarkoittavat paikallaan olevia näyttöjä, joiden läpi voi tarkastella lisätyn todellisuuden virtuaalikuvia. Tilaan sijoitetut näytöt ratkaisevat monta teknistä (valaistus), kuvanlaatuun (resoluutio) ja ihmisiin liittyviä ongelmia, mutta sovellukset näille näytöille eivät voi olla liikuteltavia [18]. Lisätyssä todellisuudessa kehitetään koko ajan myös sovelluksia ja tekniikoita, jossa ohjelmaa ohjataan keholla eikä erillisellä laitteella [19]. Näyttöjä voidaan käyttää esimerkiksi museoissa tai akvaarioissa, esimerkiksi lisäämään sukupuuttoon kuolleita lajeja [20] sekä auttamaan vaatevalinnassa [17].

2.2. Lisätyn todellisuuden sovelluksia

Lisätyn todellisuuden hyödyntäminen pelaamisessa on pitkään pohdittu idea [21]. Monet ihmiset tutustuivat lähemmin lisätyn todellisuuden mahdollisuuksiin vuonna 2016 suosituksi nousseen Pokemon Go:n myötä. kehitetään paljon erilaisia pelejä [22]. Yksi tapa käyttää lisätyn todellisuuden sovelluksia on siedätyshoito fobioihin [23]. Yksi suurimmista nousussa olevista pelaamisen alustoista ovat mobiililaitteet, joihin lisätty todellisuus toimii todella hyvin, koska mobiililaitteet ovat kevyitä ja helposti liikuteltavia. Tämän lisäksi tehokkaat mobiililaitteet ovat koko ajan yleistymään päin [15].

2.2.1. Opetuskäyttö

Opetukseen liittyviä lisätyn todellisuuden pelejä on kehitteillä jo paljon, varsinkin mobiililaitteille [24][25]. Opetuskäyttöön tutkitaan lisätyn todellisuuden sovelluksia koko ajan [26]. Opetuskäytössä lisätyn todellisuuden ajatellaan parantavan

opiskelijoiden motivaatiota, auttamaan heitä pääsemään kiinni asiaan, lisäämään opettaja-oppilas-vuorovaikutusta sekä auttamaan opiskelijoita opiskelemaan omalla tavallaan ja omaan tahtiin [27]. EU:ssa on kehitetty ”Science Center to Go” -järjestelmää, jonka tarkoituksena on tuoda lisätyn todellisuuden oppiminen jokaiseen luokkahuoneeseen [28]. ”Science Center to Go” on Euroopan unionin tukema järjestelmä, jossa lisätyn todellisuuden avulla, voidaan tutustua erilaisiin tieteellisiin kokeisiin, kuten Boltzmannin vakioon ja mekaniikkaan [29]. Järjestelmässä käytetään pelkästään salkkua, joka sisältää pelkästään web-kameran, 3D-printattavia osia, tabletin sekä käyttöohjeet. ”Science Center to Go” -järjestelmää on esitelty monissa tilaisuuksissa ympäri Eurooppaa ja se pyrkii vahvistamaan muodollista koulutusta ja arkioppimista [28]. ”Science Center to Go” -järjestelmän mielekkyydestä on tehty tutkimusta, ja tuloksien mukaan opettajat odottavat järjestelmän kehittymistä innolla sekä oppilaat ovat olleet mielissään päästessään työskentelemään järjestelmän kanssa [28]. Erilainen opetuskäyttöön ja muuhun käyttöön suunniteltu AR:n sovellus on AR-kirjat, joissa esimerkiksi asioista voidaan luoda 3D-malleja opittavista asioista [27]. Opetuskäyttöön voi kehitellä myös pelejä, kuten evoluutiosta ja luonnonvalinnasta kertova SimSnails [30] sekä Alien Contact, jossa opiskelijoiden tulee selvittää, miksi ”alienit” ovat tulleet maahan selvittämällä vihjeitä ja ratkaisemalla arvoituksia [27]. Lisättyä todellisuutta voi hyödyntää myös suoraan ympäristössä. Euroopan Unionin rahoittama iTacitus projekti voisi tuoda erilaiset historialliset tapahtumat eloon historiallisella paikalla. Tämän lisäksi Wikitude ja samankaltaiset sovellukset voivat auttaa lisätyn todellisuuden lisäämistä esimerkiksi luokkaretkille, jonka aikana kerätään esimerkiksi koodeja tai ratkaistaan pulmia [27].

2.2.2. Viihdekäyttö

Yksi ensimmäisistä viihdekäyttöön tehdyistä lisätyn todellisuuden sovelluksista on ARQuake, jossa selkään kiinnitettävän valjaan ja tietokoneen sekä muoviaseen kanssa pystyy pelaamaan oikeassa maailmassa Quakea, joka on vuonna 1996 julkaistu, tunnettu videopeli [31]. Pelissä pystyi tosin liikkumaan vain tietyllä kampuksen alueella, joka oli mallinnettu peliin, jotta esimerkiksi viholliset eivät näkyisi paikoissa, joissa niitä ei saisi näkyä. ”Vakavat pelit” ovat liikuntaa ja terveyttä suosivia pelejä, joita kehitellään muun muassa lisättyyn todellisuuteen [32].

2.2.3. Sotilaskäyttö

Monien maiden puolustusvoimat ovat kiinnostuneet lisätyn todellisuuden sovellutuksista taistelijoille. Sotilaskäyttöön on kehitetty muun muassa BARS-järjestelmää, joka on suunniteltu käytettäväksi kaupunkiolosuhteissa [33]. Huolto- ja korjaustöihin on lisäksi kehitelty omaa HMD:tä käyttävää sovellusta [34][35]. Sovelluksessa päähän kiinnitetty laite näyttää ohjeita, lähikuvia ja kertoo eri osien tarkoituksen. Tällaista lisätyn todellisuuden sovellusta voitaisiin käyttää myös muilla aloilla [36]. Lisäksi sotilaskäyttöön kehitetään erilaisia opetusjärjestelmiä, joissa lisätyssä todellisuudessa tunnistetaan ongelmia fyysisessä ympäristössä [37]. Sotilaskäyttöön suunnitellussa tutkimuksessa on keskitytty myös informaation suodattamiseen, sillä hektisessä ja vaarallisessa ympäristössä turha informaatio voi

olla vaaraksi [38]. Informaation suodattaminen on tärkeää myös muillakin aloilla, jotta informaation määrää voidaan lisätä yleisesti.

2.2.4. Teollisuuskäyttö

Teollisuudessa lisätyn todellisuuden sovellukset ovat jossakin määrin samanlaisia kuin sotilaskäytössä, esimerkiksi korjaus- ja huoltotyöt ajoneuvoille ovat samanlaisia [6][34][36]. Lisättyä todellisuutta sovelletaan myös rakennuksien mallien ja tietojen käytössä [39]. Lisätyn todellisuuden sovellutuksia on tutkittu teollisuuden eri näkökulmista [40]. Autoteollisuuteen on suunniteltu muun muassa Space Design MR-nimistä työtilaa, jossa autoa voi muokata ja visualisoida sekä virtuaalisessa autossa voi kokeilla auton tilavuutta [6]. BMW on kokeillut parantaa hitsaustyötä lisätyn todellisuuden avulla [6]. Volkswagen on myös suunnitellut tuotantolinjoja ja työpajoja sekä tarkastanut osia [6]. Lisäksi pienten robottien, kuten Roomban, reittien suunnittelussa on hyödynnetty lisättyä todellisuutta. Erilaisten rakennusprojektien putki- ja johtosuunnittelussa on hyödynnetty lisättyä todellisuutta [6].

2.2.5. Lääketieteellinen käyttö

Lisättyä todellisuutta on sovellettu myös lääketieteelliseen käyttöön. AR:a on käytetty esimerkiksi leikkauksessa kuvaamaan leikattavaa aluetta tai nukkepotilaan kanssa antamaan tuntopalautetta tutkimuksessa [6]. Lääkärit ja hoitajat voisivat myös käyttää HMD-laseja ja saada sitä kautta tarvittavat potilastiedot sekä hälytykset kiireellisiin tapauksiin [7]. Lääketieteen suurimpia sovellutuksia lisätylle todellisuudelle ovat robottien suorittama kirurginen leikkaus ja erilaiset kehon kuvantamiset. Robotin käyttäminen leikkauksissa sisältää paljon hyviä puolia, kuten pienemmät haavat ja vähemmän invasiiviset leikkaukset, mutta huonoina puolina ovat muun muassa tuntoaistin vähyys robotissa [17].

3. KUVAN MATALAN TASON PROSESSOINTI

Kappaleiden tunnistaminen kuvasta on haastava ongelma, joka voi vaatia monimutkaista kuvan prosessointia. Tässä osiossa käsitellään kuvan matalan tason prosessointia, jonka käyttäminen on edellytyksenä korkean tason operaatioille, kuten esimerkiksi jonkin ennalta määritellyn kappaleen tunnistamiselle kuvasta.

Monissa kuvia käsittelevissä sovelluksissa ensimmäisenä vaiheena on kuva segmentoitava jatkokäsittelyä varten mielekkäisiin osiin. Kuvan segmentointi on keskeinen osa monia kappaleen tunnistamiseen liittyviä sovelluksia. Kuvan segmentointi on melko vaikea ongelma ja sen hankaluus vain korostuu, mikäli segmentoinnin halutaan tapahtuvan onnistuneesti ennalta kontrolloimattomaan kuvaan tai ilman minkäänlaisia ennalta saatavia avustavia tietoja. Kuvan segmentointiin on olemassa lukuisia tekniikoita, kuten esimerkiksi kynnystäminen ja alueen kasvattaminen.

Piirteiden tunnistaminen tarkoittaa kuvan tutkimista eri metodeilla, jotta voitaisiin selvittää missä kuvan kohdassa on mitäkin piirteitä. Käytännössä kuvan jokainen pikseli ja sen paikallinen ympäristö tutkitaan valitun metodin avulla ja tehdään päätös, onko kuvan pikselissä olemassa etsittävä piirre vai ei. Kuvasta voidaan tunnistaa erilaisia piirteitä, kuten reunoja, kulmia ja yhtenäisiä alueita. Piirteiden tunnistuksen lopputuloksena on alkuperäisen kuvan pisteiden osajoukko, jonka pisteiden on katsottu omaavaan halutun piirteen. Kuvan reunojen tunnistukseen voidaan käyttää esimerkiksi Canny-reunantunnistusalgoritmia ja kulmien tunnistukseen Harris-kulmantunnistusalgoritmia.

3.1. Segmentointi

Kuvan segmentointi tarkoittaa kuvan jakamista mielekkäisiin osiin, eli toisin sanoen pikselijoukkoihin, joilla on joitakin yhteisiä piirteitä. Segmentoinnin lopputavoite on saada kuva jaettua osiin niin, että jako saa kuvan eri osat eriteltyä tarkoituksenmukaisesti ja tekee kuvasta helpommin analysoitavan, mikä on usein välttämätöntä kuvan jatkokäsittelyn onnistumiselle. Kuvan segmentointia käytetään yleensä kappaleiden ja rajojen tunnistamiseen kuvasta.

Kuvan segmentointi on haasteellinen ongelma. Kuvan segmentointiin on kehitetty runsaasti erilaisia algoritmeja, joista yleisimpiä käsitellään tässä luvussa. Kuvan segmentointiin käytettävän algoritmin valinnassa on hyödynnettävä saatavilla olevaa ennakkotietoa käsiteltävän kuvan luonteesta, jotta segmentointi olisi tehokasta. Jokainen kuvan segmentointiin perustuva algoritmi perustuu pohjimmiltaan oletukseen, että samaan kuvan alueeseen kuuluvilla pikseleillä on joitakin samankaltaisia ominaisuuksia.

Segmentointi voi olla ohjaamatonta tai ohjattua. Ohjaamaton segmentointi pyrkii jakamaan kuvan alueisiin ilman ennakkotietoja kuvan alueiden ominaisuuksista. Ohjattu segmentointi voi käyttää esimerkiksi ennalta valittuja pikseleitä kuulumaan tiettyihin alueisiin ja siten toimimaan apuna kuvan pikseleiden luokittelussa omiksi alueikseen.

Tässä työn osiossa käsitellään erilaisia menetelmiä ja algoritmeja, joita voidaan käyttää kuvan segmentointiin.

3.1.1. Kynnystäminen

Yksinkertainen, vanha ja suosittu tapa segmentoida kuva on käyttää harmaasävyn kynnysarvoa, jotta voitaisiin jakaa kuvan pikselit kahteen osaan, joita voidaan kutsua myös kuvan etualaksi, eli kohteeksi ja taustaksi [41]. Kynnysarvon valinta voidaan suorittaa etukäteen valitsemisen sijaan erilaisilla algoritmeilla, joita on kehitetty tarkoitusta varten lukuisia. Algoritmit voidaan jakaa kuuteen luokkaan [42]: histogrammin muotoon perustuvat, klusterointiin perustuvat, entropiaan perustuvat, kohteen piirteisiin perustuvat, spatiaalisuuteen perustuvat ja paikalliseen adaptoitumiseen perustuvat algoritmit.

Histogrammin muotoon perustuvia metodeja ovat esimerkiksi histogrammin konveksin verhon analyysiin perustuva metodi [43] ja Balanced Histogram Thresholding-menetelmä [44], jossa histogrammi asetetaan tasapainovaa'alle keskikohdastaan ja sille etsitään tasapainokohta poistamalla joka algoritmin kierroksella raskaammalta puolen painoa histogrammista.

Klusterointiin perustuvissa metodeissa tehdään klusterianalyysi kuvan pikseleille olettaen kuvassa olevan aina kaksi klusteria. Esimerkkejä klusterointiin perustuvasta kynnysarvon etsinnästä ovat iteratiivinen metodi [45], Otsun metodi [46], joka pyrkii minimoimaan pikseliluokkien sisäisen varianssin ja pienintä virhettä tavoitteleva metodi [47], joka olettaa objektin ja taustan harmaasävyjen olevan Gaussin jakauman mukaisia ja sumeaa klusterointia [48] hyödyntävä metodi.

Entropiaan perustuvat algoritmit hyödyntävät kuvassa olevaa harmaasävyjakauman entropiaa. Esimerkkejä entropiaan perustuvista periaatteista ovat esimerkiksi segmentoidun kuvan taustan ja kohteen entropioiden summan maksimointiin pyrkiminen [49], alkuperäisen ja binärisoidun kuvan välisen ristientropian minimointiin pyrkiminen [50] ja sumeiden joukkojen hyödyntäminen entropian laskennassa [51].

Kohteen piirteisiin perustuvat algoritmit käyttävät hyväkseen mitattavissa olevaa samankaltaisuutta alkuperäisen kuvan ja binärisoidun kuvan välillä. Tähän kategoriaan kuuluvat kynnysarvon valintaa tekevät algoritmit voivat esimerkiksi hyödyntää reunojen vastaavuutta [52] tai kuvan momenttien säilyttämistä [53].

Spatiaaliset algoritmit hyödyntävät kuvan pikseleiden harmaasävyjakauman lisäksi myös pikseleiden naapuruston tutkimista. Tässä algoritmiluokassa algoritmit voivat hyödyntää esimerkiksi harmaasävyn yhteisesiintymisen matriiseja [54].

Paikalliseen adaptoitumiseen perustuvat algoritmit laskevat jokaiselle kuvan pikselille oman kynnysarvonsa perustuen tämän pikselin naapuruston ominaisuuksiin. Nämä algoritmit voivat hyödyntää esimerkiksi naapurustonsa keskiarvoa ja varianssia [55] tai kuvan paikallista kontrastia [56].

3.1.2. K:n keskiarvon klusterointi

Kuvan segmentointiin voidaan käyttää K:n keskiarvon klusterointia [57][58]. Tällöin valitaan K määrä kohdepikseleitä kuvasta, ja asetetaan jokainen muu pikseli samaan joukkoon sen kohdepikselin kanssa, joka on väriltään lähinnä kyseistä pikseliä. Tämän jälkeen lasketaan uudet kohdepikselit ja operaatiota toistetaan niin kauan, että muutoksia uudella kierroksella ei enää tapahdu. Suurin ongelma on valita oikea suuruus K:lle [59]. K:n keskiarvon klusterointi on käyttökelpoinen työkalu ja sitä sovellettu useasti kuvan segmentointitarkoituksiin käyttämällä sitä pohjana monissa erilaisissa kuvankäsittelytilanteissa ja algoritmeissa [60][61][62][63].

3.1.3. Sumea C:n keskiarvon klusterointi

Sumea C:n keskiarvon klusterointi kehitettiin 1970-luvulla [64]. Sumea C:n keskiarvon klusterointi on hyvin läheistä sukua K:n keskiarvon klusteroinnille, mutta poikkeuksena tässä käytetään jokaiselle datapisteelle sumeaa kuuluvuutta kuhunkin klusteriin. Sumea C:n keskiarvon klusterointi on ollut suosittu menetelmä sovellettavaksi kuvan segmentointialgoritmeissa. Menetelmää on sovellettu käytettäväksi kuvan segmentointiin esimerkiksi hyödyntämällä kuvan spatiaalista informaatiota [65][66], käyttämällä spatiaalista sumeaa klusterointia pohjana Level set-segmentoinnille [67] tai hyödyntämällä kuvan paikallista informaatiota kuvan segmentoinnin tehostamiseksi [68].

3.1.4. Kompressointiin perustuva segmentointi

Kompressointiin perustuvassa metodissa [69][70] hyödynnetään havaintoa, että kuvan homogeenisten alueiden tekstuureja voi tehokkaasti mallintaa Gaussin jakaumalla. Menetelmässä hyödynnetään myös sitä, että alueiden rajoja voidaan tehokkaasti koodata ketjukoodin avulla. Menetelmä tavoittelee optimaalista segmentointia pyrkimällä saavuttamaan lyhimmän mahdollisen reunan ja tekstuurien koodauksen pituuden. Täten siis kompressointiin perustuvan menetelmän toimivuus rakentuu kokonaan taustaoletukselle, että kaikista mahdollisista kuvan segmentoinneista on paras se, joka samalla minimoi datan koodaukseen käytettävän tilan.

3.1.5. Alueen kasvatus

Alueen kasvatuksessa käytetään oletusta, että kuvan samaan alueeseen kuuluvat pikselit ovat samankaltaisia arvoiltaan. Perusperiaatteena tähän luokkaan kuuluvissa algoritmeissa on vertailla jokaista kuvan pikseliä sen naapureiden kanssa ja sen perusteella tehdä päätös, kuuluuko tarkasteltava pikseli samaan alueeseen kuin jokin sen naapureista. Alueen kasvatuksessa voidaan käyttää ennalta valittuja pikseleitä siemenpikseleinä [71]. Tiettyyn alueeseen kuuluvia pikseleitä lähdetään etsimään siemenpikselien naapurustoista ja parhaiten johonkin alueeseen sopiva pikseli liitetään tähän alueeseen. Iteratiivista prosessia toistetaan, kunnes kaikki kuvan pikselit on jaoteltu johonkin alueeseen kuuluvaksi.

Alueen kasvatus voidaan tehdä myös ilman ennalta valittuja siemenpikseleitä [72]. Tällöin toimitaan samalla tavalla kuin siemenpikseleitä käytettäessä ja valitaan aloituspaikaksi yhden pikselin kokoinen alue. Naapuripikselit tutkitaan tähän alueen kuulumisen varalta ja ne liitetään joko löydettyihin alueisiin tai tunnistamattomalle pikselille annetaan kokonaan uusi alue. Valinta alueeseen kuulumisesta tai kuulumattomuudesta tehdään ennalta valitun raja-arvon avulla. Aloituspikseli voidaan valita satunnaisesti sen vaikuttamatta merkittävästi lopputulokseen. Ongelmina siemenpikselittömässä alueen kasvatuksessa on ennalta valittavan raja-arvon valinta ja metodin havaittavissa oleva, mutta vähäinen puolueellisuus pikseleiden luokittelussa niiden alueiden hyväksi, jotka on löydetty kuvasta aikaisemmin.

3.1.6. Watershed-segmentointimenetelmä

Kuvan segmentointiin voidaan käyttää watershed-menetelmää [73]. Watershed-menetelmässä harmaasävyinen gradienttikuva ajatellaan kolmiulotteiseksi pinnaksi, jossa korkea intensiteetti tarkoittaa korkeita kohtia, ja matala intensiteetti matalia kohtia. Segmentoinnin voidaan ajatella tapahtuvan siten, että kuvan matalia kohtia täytetään vedellä samaan tahtiin ja kun vesialueet ovat nousemassa harjanteiden yli ja koskemassa toisiaan, rakennetaan kosketuskohtiin korkea muuri, joka toimii vesialueiden jakajana. Nämä vedenjakajat toimivat lopputuloksessa segmentoitujen alueiden rajoina. Watershed-algoritmissa voidaan käyttää kuvan eri kohtien merkkausta ennen segmentointia joko esimerkiksi taustaan tai kohteeseen varmuudella kuuluviksi [74]. Tämän alkutiedon soveltaminen vähentää algoritmille ominaista ylisegmentaatiota.

3.1.7. Tilastollinen alueenyhdistäminen

Tilastollinen alueenyhdistäminen on yksi alueen kasvatukseen perustuvista metodeista [75]. Tässä menetelmässä jokainen pikseli on aluksi oma alueensa. Menetelmä muodostaa pikseleistä 4-konnektiivisuuteen perustuen pareja, jotka asetetaan järjestyksessä prioriteettilistaan käyttäen esimerkiksi harmaasävyn intensiteetin erotuksen itseisarvoa kuvaamaan pikseleiden yhteensopivuutta. Lista käydään läpi järjestyksessä ja jokainen pikseli tulee vertailluksi naapureihinsa ja mahdollisesti liitetyksi niiden kanssa samaan alueeseen. Tilastollista alueenyhdistämistä on sovellettu esimerkiksi suuren resoluution kuvien segmentointiin [76][77].

3.1.8. Graafin ositukseen perustuvat menetelmät

Graafiin hyödyntämiseen perustuvissa menetelmissä kuva käsitetään graafina, jossa solmut voivat kuvata pikseleitä tai pikselijoukkoja ja viivojen painot kuvaavat samankaltaisuutta pikselien väleillä. Graafi ositetaan halutulla kriteereillä ja jokaista pilkottua osaa kohdellaan omana alueenaan. Graafiin pohjautuvalla segmentoinnilla on useita hyviä puolia, kuten että graafiin pohjautuminen antaa täsmällisen matemaattisen rakenteen kuvalle ja lisää tehokkuutta koneelliseen laskentaan [78]. Tapoja hyödyntää graafien osittamista kuvan segmentoinnissa ovat esimerkiksi normalisoidut leikkaukset [79], pienimmän virittävän puun hyödyntäminen [80][81] ja Random Walker-algoritmi [82].

3.2. Piirteiden tunnistaminen ja erottaminen

Piirteiden tunnistaminen tarkoittaa kuvan prosessoinnin yhteydessä kuvan sisältämän informaation tutkimista, jotta saataisiin selville mitä piirteitä kuvassa on ja missä kohtaa kuvaa ne sijaitsevat. Kuvan piirteellä tarkoitetaan kuvankäsittelyn yhteydessä mitä tahansa kuvan sisältämää ominaisuutta, joka on relevantti käsillä olevan tehtävän kannalta. Piirre voi olla esimerkiksi kuvassa oleva piste, reuna tai alue.

Piirteitä tunnistavat algoritmit käyvät kuvan läpi ja tekevät paikallisesti päätöksiä onko tutkituissa pikseleissä jokin määrätty piirre vai ei. Piirteiden tunnistaminen on

matalan tason kuvankäsittelyoperaatio ja se usein suoritetaan kuvia prosessoivissa sovelluksissa ensimmäisenä operaationa, jotta käsiteltävästä kuvasta saadaan heti tunnistettua halutut piirteet jatkokäsittelyä varten.

Tässä työn osiossa käsitellään useita piirteiden tunnistamiseen käytettäviä algoritmeja.

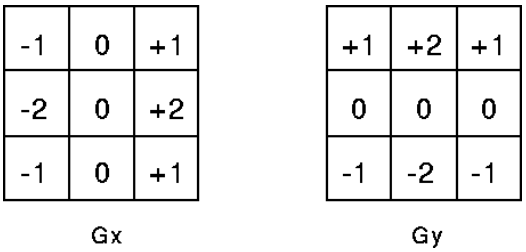
3.2.1. Canny-reunantunnistusalgoritmi

Canny on suosittu reunojen tunnistukseen käytettävä algoritmi [83]. Algoritmi noudattaa seuraavia vaiheita: Gaussin pehmentävän suodatuksen käyttö kuvan kohinan vähentämiseen, gradienttikuvan laskeminen, reunan ohentaminen, reunapikseleiden jako vahvoihin ja heikkoihin kahden raja-arvon avulla ja lopullinen reunan selvittäminen hylkäämällä heikot reunapikselit, jos ne eivät ole yhteydessä vahvoihin reunapikseleihin. Canny on havaittu suorituskyvyltään erittäin hyväksi algoritmiksi verrattuna muihin reunantunnistusalgoritmeihin lähes kaikissa tilanteissa [84][85], mutta toisaalta sen suorituskyky riippuu paljon algoritmin parametreina saamista raja-arvoista [84]. Canny on suorituskykyisempi kohinaisissa olosuhteissa verrattuna esimerkiksi Sobel-, Prewitt- ja Roberts's Cross-operaattoreihin [85].

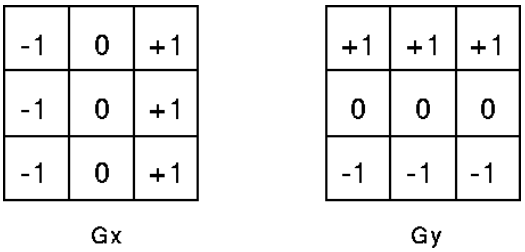
Cannyn työn pohjalta on kehitetty Deriche-reunantunnistusalgoritmi [86]. Deriche toimii samoin kuin Cannyn reunantunnistusalgoritmi, mutta se hyödyntää toisenlaista reunan tunnistamista Sobel-operaattorin sijaan ja kuvan suodattamiseen IIR (Infinite Impulse Response) suodatusta Gaussin suodatuksen sijaan.

3.2.2. Sobel, Prewitt ja Robert's Cross-operaattorit

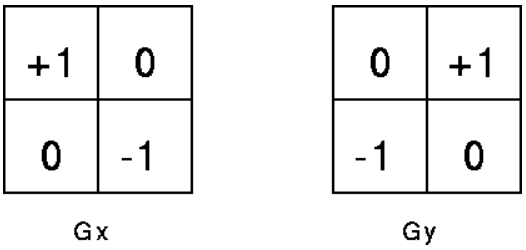
Sobel-, Prewitt-, ja Robert's Cross-operaattoreita [87][88][89] (kuvat 4, 5 ja 6) käytetään pikselien intensiteetin muutoksien tutkimiseen verrattuna pikseleiden naapurustoihin. Koska kuvan reunan kohdalla kuvan pikselien intensiteetti yleensä muuttuu voimakkaasti, on tällä keinoin mahdollista tunnistaa kuvasta reunojen olinpaikat. Nämä operaattorit ovat yksinkertaisia ja tehokkaita käyttää, mutta ne ovat herkkiä kohinalle [85]. Näillä operaattoreilla voidaan laskea jokaiselle kuvan pisteelle vaaka- ja pystysuuntaisen gradientin suuruus. Näistä arvoista voidaan selvittää kuvan pisteen kokonaisgradientin suuruus ja suunta. Mitä suurempi on kuvan pikselin saama gradientin arvo, sitä varmemmin kyseessä on kuvassa esiintyvä reuna.



Kuva 4. Sobel-operaattorin maski.



Kuva 5. Prewitt-operaattorin maski.



Kuva 6. Robert's Cross-operaattorin maski.

3.2.3. Hough-muunnos

Hough-muunnos on tekniikka piirteiden erottamiseen kuvasta. Hough-muunnoksen avulla voidaan tunnistaa kuvasta haluttuja muotoja. Ensimmäinen versio kehitettiin tunnistamaan kuvasta suoria ja kaaria [90] ja myöhemmin kehitettiin algoritmi tunnistamaan mitä tahansa määriteltyjä muotoja [91]. Probabilistinen Hough-muunnos pyrkii vähentämään Hough-muunnoksen vaatimaa laskentatehoa käyttämällä vain osaa kuvan pikseleistä hyväkseen laskennassa [92].

3.2.4. Harris-kulmantunnistusalgoritmi

Harris-kulmantunnistusalgoritmi [93] hyödyntää kuvan intensiteetin muutosta kuvan kulmien ja reunojen sijaintien tutkimisessa. Perusideana on, että käyttämällä ikkunaa kuvan jokaisen kohdan tutkimiseen pyritään löytämään kuvasta kohtia, joissa liikuttamalla ikkunaa kaikkiin suuntiin kuvan intensiteetti muuttuu voimakkaasti. Esimerkiksi tasaisella alueella ikkunan liikuttaminen ei tuota muutosta, reunan kohdalla muutosta ei tapahdu reunan suuntaisesti liikkumalla ja kulman kohdalla muutosta tapahtuu liikkuessa jokaiseen suuntaan.

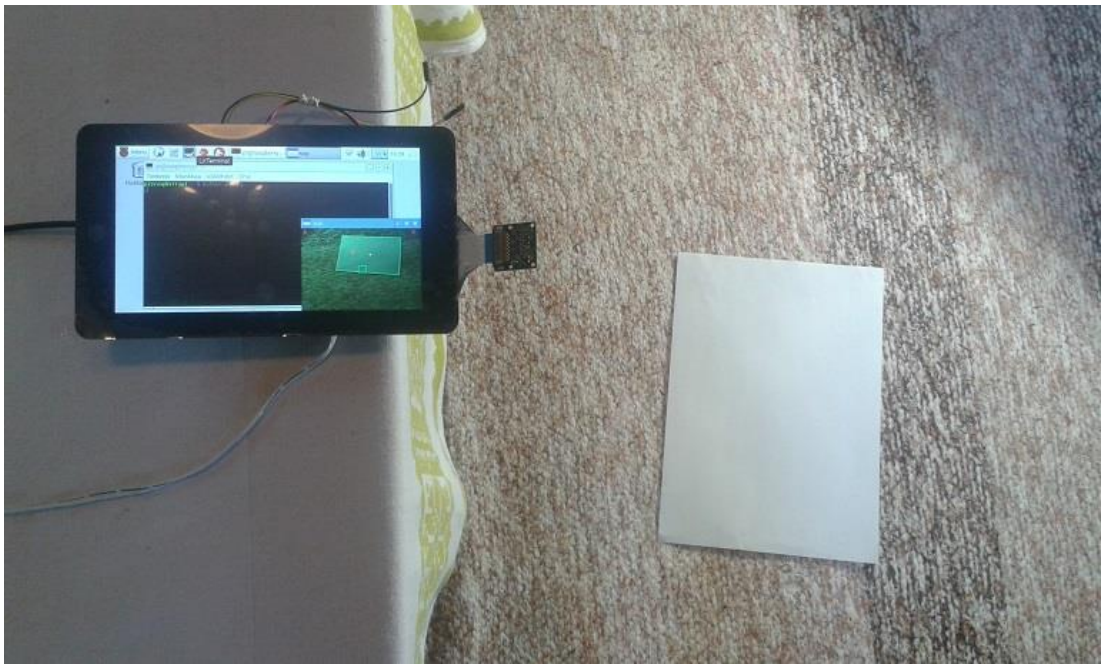
3.2.5. SUSAN-kulmantunnistusalgoritmi

SUSAN-kulmantunnistusalgoritmissa [94] käytetään ympyrän muotoista maskia jokaiseen kuvan pikseliin ja tutkitaan maskin sisällä, kuinka suuri osa maskin alueesta kuuluu samankaltaiseen segmenttiin maskin keskipisteen kanssa. Mitä pienempi on tähän segmenttiin kuuluvien pisteiden määrä ja mitä kauempana segmentin keskiö on maskin keskipisteestä, sitä todennäköisemmin kyseessä on kulma. SUSAN-kulmantunnistusalgoritmi toimii erittäin hyvin kohinaisissa olosuhteissa verrattuna muihin kulmantunnistusalgoritmeihin. Testeissä SUSAN-

algoritmin suoritusaika on vain noin yksi kymmenesosa Canny-algoritmin suoritusajasta.

4. TYÖN KUVAUS

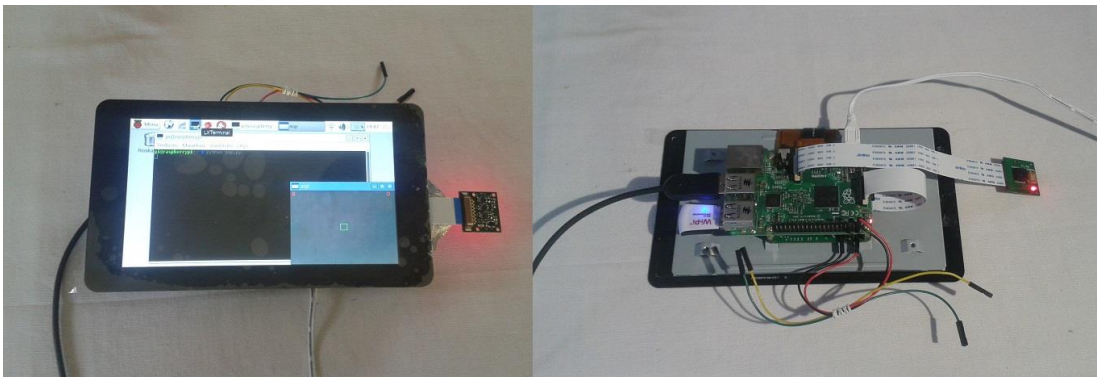
Työ on tehty Raspberry Pi 2 tietokoneelle hyödyntäen Raspberry Pi:n kameraa ja kosketusnäyttöä (kuva 7). Työssä käytetään Python ohjelmointikieltä, OpenCV- ja NumPy-ohjelmointikirjastoja.



Kuva 7. Tässä työssä luotu AR-sovellus tunnistaa A4-paperiarkin.

4.1. Laitteisto ja ohjelmisto

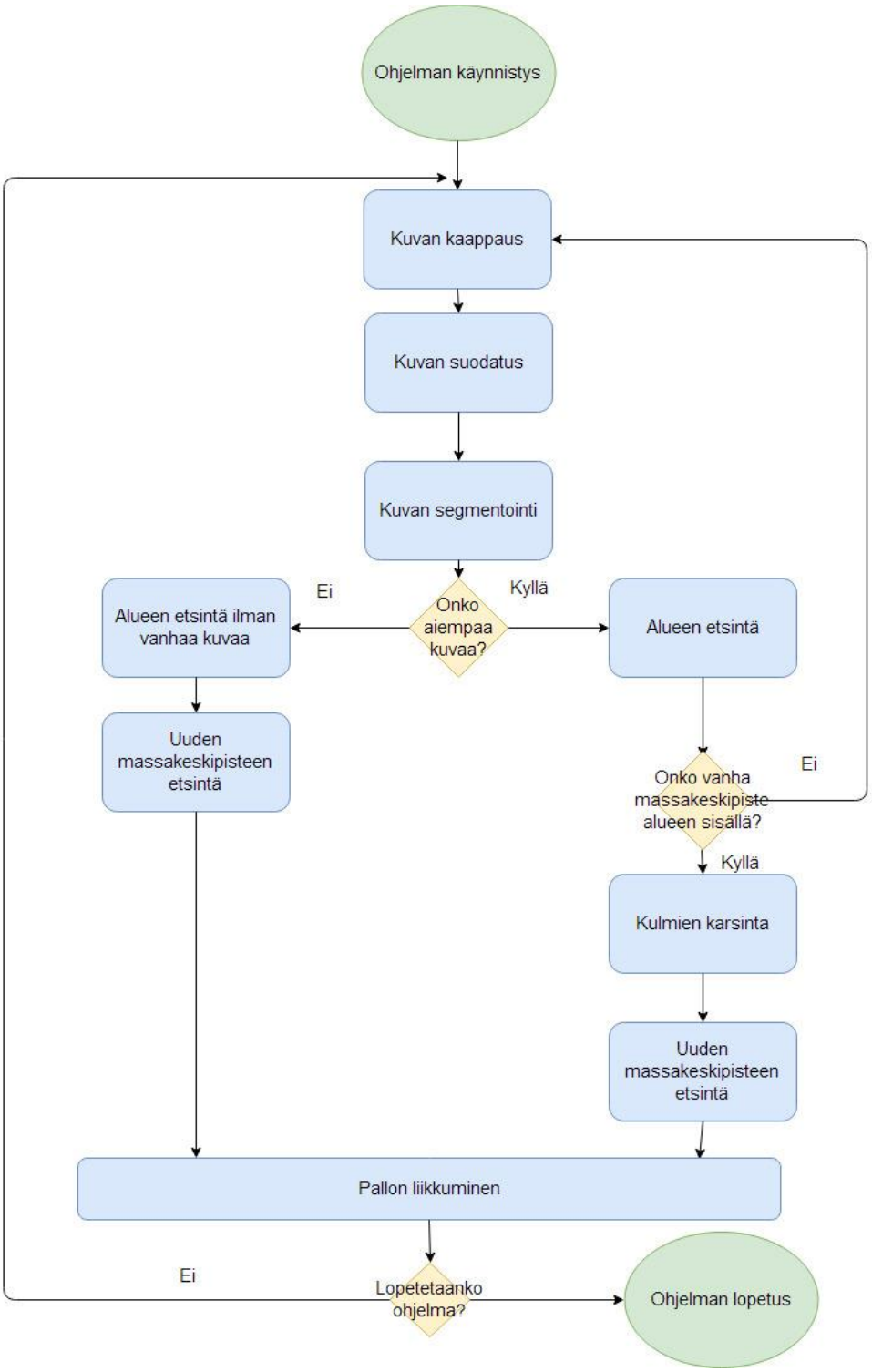
Työssä käytettiin Raspberry Pi 2 minitietokonetta (kuva 8), jossa käyttöjärjestelmänä toimi Rasbian 8.0. Minitietokoneessa oli 1Gt RAM-muistia sekä neliytiminen 900MHz:n ARM Cortex A7 prosessori [95]. Minitietokoneeseen oli liitetty myös USB portteihin WLAN lähetin-vastaanotin sekä näppäimistö ja hiiri. Minitietokoneen CSI porttiin oli liitetty Raspberry PI Camera Module v2. Kameralla pystyi kuvaamaan 1080p30 sekä 720p60 videota [95]. Raspberry Pi2 minitietokone käyttää 7 tuuman Raspberry Pi Display kosketusnäyttöä, joka kiinnitettiin DSI- sekä GPIO-portteihin. Näytönresoluutio on 800x480 [95]. Työssä käytettiin Pythonin versiota 2.7.9 ja PiCamera kirjaston versiota 1.12. Kirjastoa käytettiin kameran alustamiseen sekä kuvankaappaukseen. Lisäksi työssä käytettiin Numpya, joka on Pythonin tieteellisen laskennan kirjasto. Kirjasto keskittyy monimutkaiseen laskentaan sekä suuriin taulukoihin [96]. Työssä käytettiin myös OpenCV:tä, joka on ilmainen ohjelmistokirjasto. Kirjasto keskittyy tarjoamaan laskennallista tehokkuutta sekä konenäön että kuvan muokkauksen osalta [97]. Työssä käytettiin OpenCV:n kirjastoversiota 2.4.13 sekä NumPy:n kirjastoversiota 1.8.2.



Kuva 8. Työssä käytetty Raspberry Pi 2.

4.2. Ohjelman kuvaus

Ohjelman on tarkoitus löytää pelialue, johon virtuaalinen pallo sijoitetaan ja jossa pallo pystyy liikkumaan menemättä rajojen ulkopuolelle. Ohjelman toiminta perustuu silmukkaan (kuva 9), joka alkaa kuvan kaappaamisesta. Sen jälkeen kuva suodatetaan ja seuraavaksi kuva segmentoidaan. Segmentoidusta kuvasta etsitään pelialue ja siihen lisätään pallo.



Kuva 9. Ohjelman toiminta.

4.2.1. Kuvien kaappaaminen

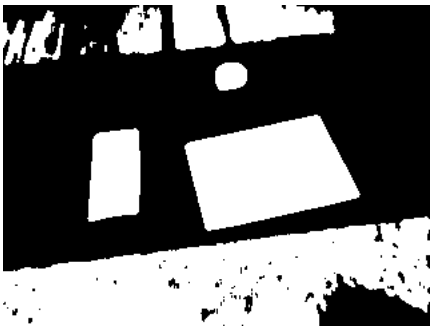
Kuvat kaapataan 320x240 pikselin resoluutiolla. Tätä suurempi resoluutio olisi laskennallisesti raskas, eikä Raspberry Pi:n teho riittäisi kuvien reaaliaikaiseen käsittelyyn. Kun Raspberry Pi:n kamera on kaapannut kuvan, niin sen jälkeen se muutetaan numpy-taulukoksi, jotta kuvaa olisi helppo käsitellä ohjelmassa. Kuvan kaappaamisen jälkeen se muutetaan harmaasävyiseksi, koska harmaasävyinen kuva on parempi jatkokäsittelyä varten.

4.2.2. Kuvien suodatus

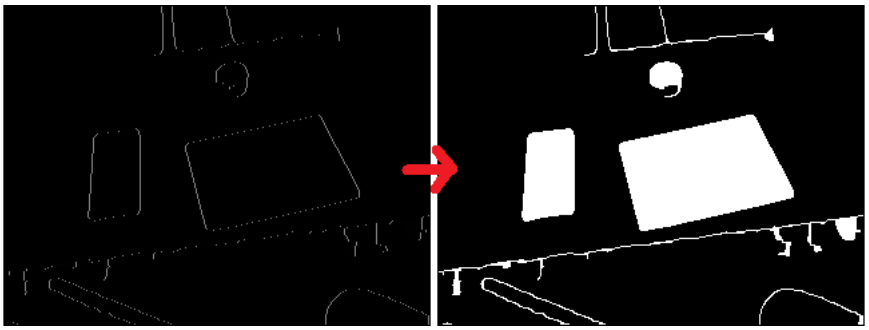
Kuvien kaappaamisen jälkeen ne suodatetaan. Ohjelmassa harmaasävyiset kuvat voidaan suodattaa käyttäen neljää erilaista suodatinta: 1. keskiarvosuodatus, 2. Gaussin suodatus 3. mediaanisuodatus 4. bilateraalinen suodatus. Suodattimissa käytetään 5x5-kokoista maskia. Tätä suurempi maski olisi laskennallisesti raskas, eikä näin ollen sovellu hyvin reaaliaikasovelluksiin.

4.2.3. Kuvien segmentointi

Ohjelma voi segmentoida kuvan kahdella eri tavalla. Ensimmäinen tapa käyttää Otsun metodia (kuva 10). Tämä tapa toimii tietyissä tilanteissa hyvin, mutta jos pelialueen väri ja taustan väri ovat lähellä toisiaan, niin Otsun metodi ei löydä aluetta oikein. Toinen tapa käyttää Canny-reunantunnistusta. Cannyn parametreina käytetään arvoja 10 ja 200. Valitsemalla pieni ensimmäinen parametri varmistetaan, että algoritmi luokittele pelialueen reunapikselit vähintään heikoiksi reunapikseleiksi. Samalla algoritmi löytää muita heikkoja reunapikseleitä, jotka eivät kuulu pelialueen reunaan, mutta näistä ei ole haittaa ohjelman toiminnalle. Tämän jälkeen molemmissa tavoissa kuvasta etsitään yhtenäiset alueet OpenCV:n findContours-funktiolla. Funktio etsii yhtenäiset alueet seuraamalla alueen reunapikseleitä [98]. Tällä tavalla myös Canny-reunantunnistus jakaa kuvan alueisiin, eikä pelkästään reunoihin (kuva 11). Jotta Canny löytää alueen oikein, täytyy alueen reunan olla yhtenäinen. Ohjelmassa reunan yhtenäisyyttä parannetaan tekemällä morfologinen sulkeminen kuvaan.



Kuva 10. Otsun metodilla tehty segmentointi.



Kuva 11. Canny-reunantunnistuksesta kuva saadaan segmentoitua yhtenäisiksi alueiksi findContours-funktiolla.

4.2.4. Pelialueen etsiminen

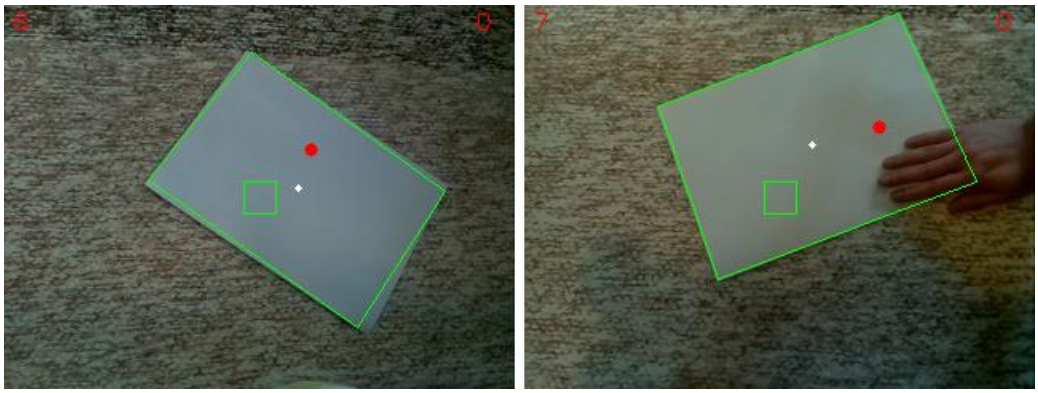
Pelialueen etsintä toimii kahdella eri tavalla; joko alue on löydetty jo aikaisemmin tai aluetta ei ole löydetty aikaisemmin. Kun pelialuetta ei ole löytynyt aikaisemmin niin ohjelma toimii seuraavasti: Segmentoinnin tuloksena löytyneistä alueista poistetaan turhat kulmapisteet, jotka ovat alueiden reunoissa. Tämä tehdään OpenCV:n approxPolyDP-funktiolla, se poistaa kulmapisteet, joiden poistaminen ei muuta aluetta paljon. Tämän jälkeen ohjelma etsii suurimman alueen, jossa on neljä noin 90:n asteen kulmaa. Jos tämän kaltainen alue löytyy, todetaan sen olevan pelialue.

Kun pelialue on löytynyt ensimmäisen kerran, niin ohjelman toiminta vaihtuu toiseen osioon. Tästä eteenpäin ohjelma tekee alueisiin konveksin verhon (convex hull), jotta alueista häviävät sisäänpäin kääntyneet osat. Esimerkiksi paperin päällä oleva käsi tekee tällaisen alueen. Seuraavaksi alueisiin tehdään turhien kulmapisteiden poisto approxPolyDP-funktiolla (kuva 12). Tämän jälkeen pelialueeksi valitaan se alue, jonka alueessa edellisen kuvan pelialueen keskipiste sijaitsee. Tässä vaiheessa pelialueeseen on jäänyt yleensä 4-5 kulmapistettä.

Kuva 12. Esimerkki pelialueen turhien kulmapisteiden poistosta. Aluksi alueelle tehdään konveksi verho. Ja tämän jälkeen jäljelle jääneet turhat kulmat poistetaan approxPolyDP-funktiolla. Punaiset pisteet ovat alueen kulmapisteitä.

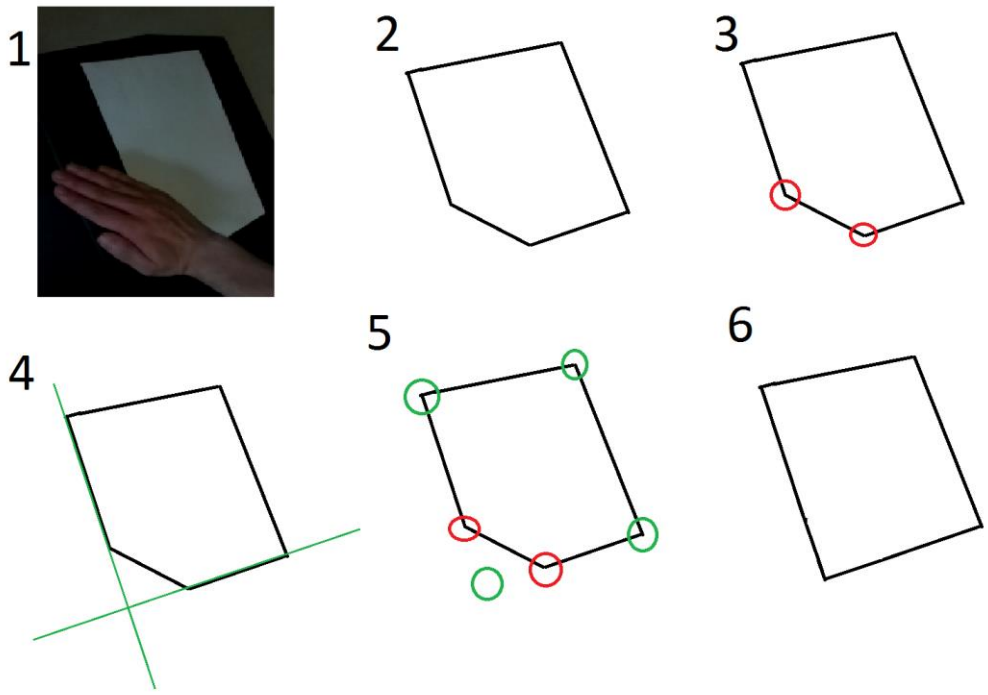
Jos aluetta ei ole löytynyt aikaisemmin, niin pelialueen löytämisen jälkeen ei tarvita jatkotoimenpiteitä, koska tässä vaiheessa tiedetään varmasti, että algoritmi löysi vain 4 kulmaa. Sitä vastoin, jos pelialue on jo löytynyt aikaisemmin, niin ohjelma valitsee pelialueen edellisen kuvan perusteella. Tässä vaiheessa on mahdollista, että käsi on pelialueen päällä (kuva 13). Jos käsi ei ole pelialueen päällä, niin jatkotoimenpiteitä

ei tarvita, koska algoritmi löytää 4 kulmaa. Jos jokin pelialueen kulmista on peitetty kädellä, niin edellä esitetyt vaiheet tuottavat viisikulmaisen alueen. Tässä tapauksessa suoritetaan uuden kulman paikan arviointi ja karsitaan pois ne pisteet, jotka eivät ole pelialueen kulmia.



Kuva 13. Löytynyt pelialue (vihreä viiva), keskipiste (valkoinen piste) ja pallo (punainen piste).

Edellä mainittu prosessi toimii seuraavasti: Aluksi alueesta tunnistetaan heikot kulmat yksinkertaisesti vertailemalla kaikkien kulmien suuruuksia ja merkitsemällä suurimmat kulmat heikoiksi. Toisessa vaiheessa asetetaan suorat kulkemaan alueen reunan suuntaisesti heikkojen kulmapisteiden kautta. Jos asetetut suorat kohtaavat toisensa, on löytynyt uusi kulmapiste. Viimeisessä vaiheessa alkuperäisistä ja löydetyistä uusista pisteistä valitaan ne neljä kulmapistettä, joiden keskimääräinen etäisyys kaikkiin muihin kulmapisteisiin on suurin. Prosessin vaiheet on havainnollistettu piirroksin kuvassa 14.

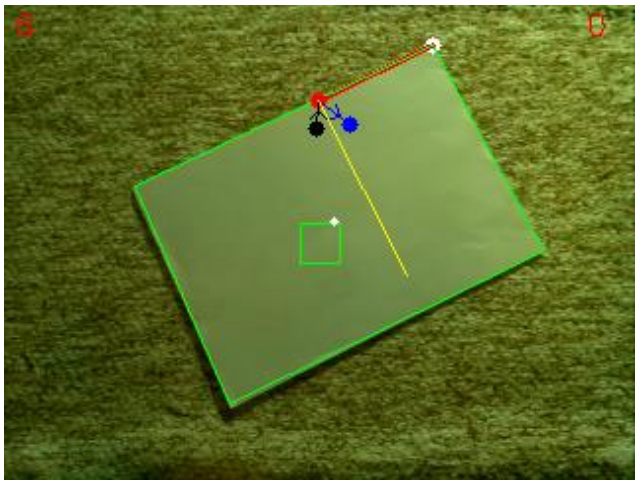


Kuva 14. Peitetyn kulman paikan arvioinnin vaiheet. Kohdassa 1 alkuperäinen kuva. Kohdassa 2 löydetty yhtenäinen alue. Kohdassa 3 heikkojen kulmien tunnistaminen. Kohdassa 4 suorien asettaminen. Kohdassa 5 suurimman keskietäisyyden kulmapisteiden valitseminen. Kohdassa 6 prosessin lopputulos.

4.2.5. Pallon liikkuminen

Pallon sijainti lasketaan verrattuna pelialueen keskipisteeseen. Tällä tavalla pallo pysyy oikeassa paikassa suhteessa pelialueeseen, vaikka kameraa liikutettaisiin. Ohjelman alussa pallo on pelialueen keskipisteessä, josta se alkaa liikkua satunnaiseen suuntaan (pallon suunta on kulma joka voi saada arvoja välillä $-\pi - \pi$). Jokaisella ohjelman syklillä pallo liikkuu pelialueella ennalta määritellyn matkan. Pallo liikkuu tämän matkan kerralla. Jos pallo on vielä liikkumisen jälkeen pelialueessa, ohjelma jatkuu eteenpäin normaalisti. Mutta tilanteessa jossa pallo on ylittämässä pelialueen reunan, pallo ei pysähdy pelialueen reunaan, vaan se hyppää reunan yli pelialueen ulkopuolelle. Kun tällainen tilanne tapahtuu, niin pallo liikkuu päinvastaiseen suuntaan pikseli kerrallaan, kunnes pallo on palannut pelialueeseen. Tässä tilanteessa tiedetään, että pallo on pelialueen reunassa ja sen pitäisi kimmota reunasta uuteen suuntaan.

Tämän jälkeen etsitään reuna, jonka suhteen pallo kimpoaa. Se löydetään etsimällä lähin kulmapiste ja muodostamalla viiva pallon ja lähimmän kulmapisteen välille (kuva 15). Sitten lasketaan tämän viivan suunta välillä $-\pi - \pi$. Jonka jälkeen voidaan laskea pallon kimpoamisen jälkeinen suunta kaavalla: ”uusi pallon suunta” = ”reunan suunta”*2 − ”pallon suunta ennen kimpoamista”. Tämän kaavan tulos voi saada arvoja, jotka ovat alle $-\pi$ tai yli π, mutta suunta on oikea kunhan aste muutetaan välille $-\pi - \pi$ lisäämällä tai vähentämällä tuloksesta 2π:n monikertoja. Tämän laskelman suorittaminen takaa sen, että pallo näyttää kimpoavan realistisesti.



Kuva 15. Esimerkki pallon (punainen piste) kimpoamisesta yläreunasta. Punainen viiva näyttää lähimmän kulman ja seinän, jonka suhteen pallon kimpoaminen lasketaan. Musta piste näyttää missä pallo oli edellisessä kuvassa. Sininen piste näyttää mihin pallo meni kimpoamisen jälkeisessä kuvassa.

5. TESTAUS

Pelialueen tunnistaminen on luodussa sovelluksessa ensiarvoisen tärkeää, koska koko muun ohjelman toiminta rakentuu pelialueen oikeellisen tunnistuksen ympärille. Lisätyn todellisuuden sovellusten on oltava toimintavarmoja erilaisissa olosuhteissa ja erilaisten häiriötekijöiden alaisena. Edellä mainittujen syiden johdosta valikoitui testien ensisijaiseksi kohteeksi pelialueen tunnistavan algoritmin suorituskyvyn testaus vaihtelevien olosuhteiden ja häiriöiden alaisena.

Lisätyn todellisuuden sovelluksissa on huomioitava, että usein laitteiston rajoitteiden takia on käytössä normaalia matalampi suorituskyky. Tämän vuoksi testattiin myös pelialueen tunnistavan algoritmin suoritusaikojen pituutta erisuuruisia kuvia käyttämällä.

Pelialueen tunnistavan algoritmin testaukseen käytettiin kuutta erilaista testiä:

- Testi 1 – Tarkkuus hyvissä olosuhteissa
- Testi 2 – Tarkkuus kulma peitettynä
- Testi 3 – Tarkkuus hämärissä olosuhteissa
- Testi 4 – Suoritusaika eri resoluution kuvilla
- Testi 5 – Tarkkuus reuna peitettynä
- Testi 6 – Tarkkuus pelialue kallistettuna

Testien tuloksissa esitetään algoritmin suorituskyky ja suoritusajat käyttäen kahta erilaista segmentointiasetusta.

5.1. Testisuunnitelma

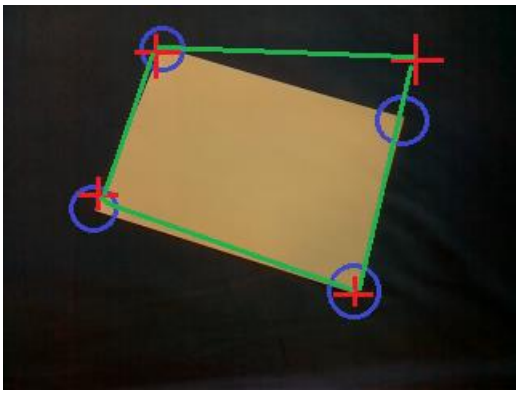
Viidessä testissä mitattiin algoritmin tarkkuutta pelialueen kulmien paikannuksessa erilaisissa olosuhteissa ja eri alkuasetuksia käyttäen. Yhdessä testissä mitattiin algoritmin suoritusaikoja eri kokoisilla kuvilla ja eri alkuasetuksilla. Testikuvamateriaali luotiin itse käyttäen rakennetun AR-laitteen kameraa ja kulmien paikoista muodostettiin pohjatotuus ihmisen arviointikykyä käyttämällä. Testeistä kaikki paitsi yksi suoritettiin tavallisella kannettavalla tietokoneella esitellyn AR-laitteen sijaan.

5.1.1. Mitattavat kohteet

Testeissä mitattiin pelialueen tunnistukseen käytettävän algoritmin toimivuutta erilaisissa olosuhteissa ja erilaisilla asetuksilla.

Pelialueen tunnistavan algoritmin saama syöte oli laitteen kameran kuvaama kuva ympäristöstä ja pelialueesta. Algoritmin tuottama tulos oli enintään neljä kappaletta kuvan koordinaattipisteitä, joissa algoritmi on tunnistanut pelialueen kulmien olevan.

Jokainen algoritmin palauttama kulma voidaan luokitella kahteen luokkaan: Onnistunut tai virheellinen tunnistus. Luokittelu tehdään vertaamalla algoritmin palauttamaa tulosta pohjatotuuteen. Jos algoritmin palauttaman kulman etäisyys pohjatotuuden kulmasta on tietyn virhemarginaalin sisällä, niin luokitellaan tunnistus onnistuneeksi. Esimerkki algoritmin palauttamien kulmien luokittelusta esitetään kuvassa 16.



Kuva 16. Esimerkki tilanteesta, jossa on tunnistettu oikein 3 kulmaa ja väärin yksi kulma. Kulmien tunnistuksien paikat on merkattu punaisilla risteillä. Oikeiden kulmien paikat ja hyväksyttävät virhemarginaalit ovat merkattu sinisillä ympyröillä.

Tutkittaessa algoritmin tarkkuutta ovat kiintoisia tietoja oikein tunnistettujen kulmien määrä, väärin tunnistettujen kulmien määrä ja väärin tunnistettujen kulmien etäisyys pohjatotuuden kulmista. Nämä tiedot raportoidaan testien tuloksissa.

Algoritmin suoritusaika mitattiin käyttämällä erikokoisia kuvia ja eri algoritmin alkuasetuksia. Ajanotto suoritettiin käyttämällä Pythonin timeit-kirjastoa ja algoritmi suoritettiin 1000 kertaa, jotta voitaisiin tuloksista laskea luotettava keskiarvo.

Algoritmia testattiin kahdella erilaisella asetuksella, jotka ovat nähtävissä taulukossa 1. Vaikka algoritmia testattiin vain yksittäisillä kuvilla, niin testiohjelma kutsui algoritmia oletuksella, että edellisestä kuvasta olisi löytynyt pelialue. Tämä tarkoittaa, että algoritmille annettiin alkutiedoksi myös yksi kuvan piste, joka varmuudella kuului pelialueeseen.

Taulukko 1. Testauksessa käytetyt algoritmin asetukset.

Alkuasetukset	**Suodatus**	**Segmentointi**
Asetukset 1 (Canny)	5x5 Mediaanisuodatus	Canny
Asetukset 2 (Otsu)	5x5 Mediaanisuodatus	Otsun metodi

5.1.2. Testit

Pelialueen tunnistavan algoritmin testaukseen käytettiin kuutta erilaista testiä. Tässä osiossa viitataan testimateriaaleihin, joista kerrotaan lisää tämän työn kohdassa 5.1.3.

Testissä 1 oli tarkoituksena mitata algoritmin toimintaa hyvissä olosuhteissa. Testin tarkoituksena oli varmistaa, että algoritmi toimii olosuhteissa, joissa ei ole mitään häiriöitä, kuten peittäviä kappaleita, huonoa valaistusta tai huonoa kuvauskulmaa pelialueeseen. Testissä käytettiin testimateriaalia 1(320x240px).

Testissä 2 peitettiin yksi pelialueen kulmista. Testin tarkoituksena oli simuloida kontrolloidusti tilannetta, jossa pelaajan käsi peittää pelialueen kulman ja siten saada tietoa algoritmin käyttäytymisestä tässä olosuhteessa. Testissä käytettiin testimateriaalia 2.

Testissä 3 käytettiin erittäin hämärissä olosuhteissa otettuja kuvia. Testin tarkoituksena oli testata algoritmin toimintaa tilanteessa, jossa valaistusolosuhteet ovat heikot. Testissä käytettiin testimateriaalia 3.

Testissä 4 käytettiin kolmea kuvasarjaa, jotka olivat täsmälleen samasta tilanteesta otettuja, mutta joiden tarkkuus oli eri. Testin tarkoituksena oli tarkkailla algoritmin suoritusaikaa erikokoisilla kuvilla. Testissä käytettiin testimateriaaleja 1(320x240px), 1(640x480px) ja 1(1280x960px).

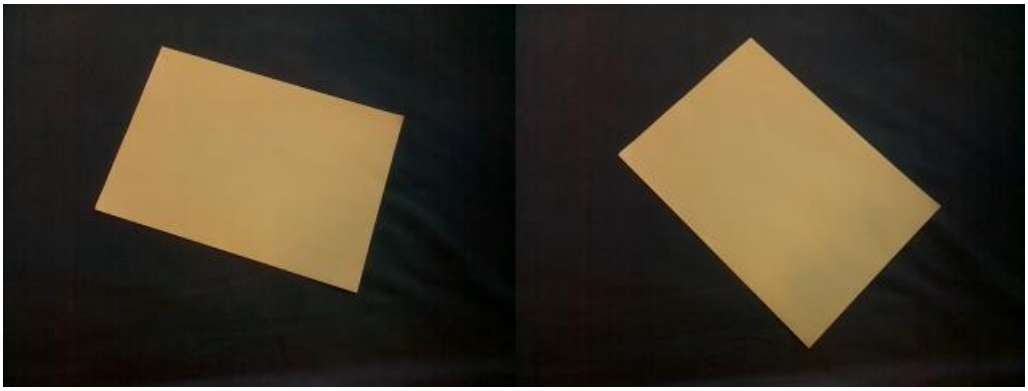
Testissä 5 peitettiin yksi pelialueen reunoista osittain. Tämä testi vastaa tilannetta, jossa pelaajan käsi on pelialueen reunan päällä. Testin tarkoituksena on selvittää kontrolloiduissa olosuhteissa, kuinka reunan peittävän käden tuoma häiriö vaikuttaa algoritmin toimintaan. Testissä käytettiin testimateriaalia 4.

Testissä 6 pelialuetta kallistettiin kameraan nähden. Tämän testin tarkoituksena on selvittää, kuinka algoritmi käyttäytyy kameran kuvauskulman ollessa jotakin muuta kohtisuora pelialueeseen nähden. Testissä käytettiin testimateriaalia 5.

5.1.3. Testimateriaali

Testimateriaali 1(320x240px) (kuva 17) sisältää 8 kuvaa hyvistä olosuhteista. Kuvissa pelialue on peittämätön, hyvin valaistu ja kamera on sijainniltaan suorassa kulmassa pelialueeseen nähden.

Testimateriaali 1(640x480px) ja 1(1280x960px) ovat täysin yhtenevät testimateriaalin 1(320x240px) kanssa, mutta niissä kuvien tarkkuus on 640x480 ja 1280x960. Kaikkien muiden testimateriaalien kuvien tarkkuus on 320x240.



Kuva 17. Testimateriaali 1(320x240px):n kuvat 2 ja 3.

Testimateriaali 2 (kuva 18) sisältää 8 kuvaa pelialueesta, jonka kulman päällä on este. Kuvasarjan edetessä estettä siirretään peittämään yhä enemmän alueen päälle peittämään yhä suurempaa osaa alueesta. Kuvasarjan ensimmäisessä kuvassa este ei mene pelialueen kulman päälle.

Kuva 18. Testimateriaali 2:n kuvat 3 ja 4.

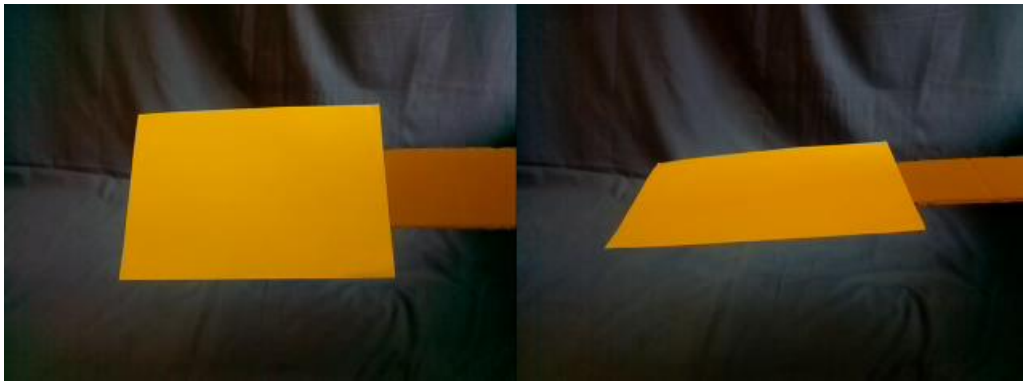
Testimateriaali 3 (kuva 19) sisältää 8 kuvaa hämärissä olosuhteissa. Kuvasarjan alussa pelialue on hyvin heikosti kuvasta erottuva. Kuvasarjan edetessä muuttuu valaistus vähitellen kirkkaammaksi ja pelialue alkaa erottua helposti nähtäväksi.

Kuva 19. Testimateriaali 3:n kuvat 3 ja 5.

Testimateriaali 4 (kuva 20) sisältää 8 kuvaa pelialueesta, jonka reunan yli on asetettu este. Kuvasarjan edetessä este siirtyy yhä enemmän reunan yli peittäen alueesta yhä suuremman osan. Ensimmäisessä kuvassa este ei ole alueen päällä. Viimeisessä kuvassa este halkaisee pelialueen kokonaan kahtia.

Kuva 20. Testimateriaali 4:n kuvat 3 ja 5.

Testimateriaali 5 (kuva 21) sisältää kuvia kallistetusta pelialueesta. Kuvasarjan alussa pelialue on kohtisuoraan kameraan nähden, mutta viimeisissä kuvissa se on kallistunut lähes vaakatasoon kameraan nähden.



Kuva 21. Testimateriaali 5:n kuvat 2 ja 4.

5.1.4. Pohjatotuuden määrittäminen

Pohjatotuudeksi valittiin jokaiselle kuvalle erikseen asetettu neljän koordinaattipisteen joukko, jossa jokainen koordinaattipiste vastaa yhtä pelialueen kulmaa. Pohjatotuus muodostettiin merkkaamalla silmämääräisesti testikuvasta neljä kulmapikseliä. Tämän jälkeen merkattu kuvamateriaali annettiin yksinkertaisen ohjelman prosessoitavaksi ja tuloksena oli jokaiselle kuvalle erikseen laskettu neljän koordinaattipisteen pohjatotuus pelialueen kulmapisteiden paikasta.

Ihmisen luoman pohjatotuuden ja algoritmin tuottamien tulosten välinen vertailu valittiin evaluointimenetelmäksi, koska testeissä käytetyt kuvat ovat yksinkertaisia ja kuvan kiinnostuksen kohteena toimivat pelialueen kulmat ovat selkeästi ja nopeasti ihmisen nähtävissä.

Ongelmaksi muodostui, että ihmisen asettama pohjatotuus pelialueen kulmien paikasta on aina subjektiivinen näkemys. Tämä johti testitulokset näyttämään pientä epätarkkuutta niissäkin tapauksissa, joissa algoritmin tunnistamat pelialueen kulmien paikat olivat aivan hyväksyttäviä käytännön sovelluksia varten. Koska absoluuttista pohjatotuutta ei voitu luoda, ratkaisuksi valittiin hyväksyä virheiden olemassaolo ja luokitella tietyn virhemarginaalin sisällä tapahtuva pelialueen kulman sijainnin tunnistus onnistuneeksi kulman tunnistamiseksi.

5.1.5. Toteutus

Testit 1, 2, 3, 5 ja 6 pelialueen tunnistavalle algoritmille suoritettiin rakennetun AR-laitteen sijaan kannettavalla tietokoneella. Algoritmin tuottamien tulosten kannalta tällä ei ollut mitään merkitystä, koska algoritmin lopputulos on riippuvainen ainoastaan sen saamasta kuvasta eikä laitteiston ominaisuuksista. Käytetty kannettava tietokone oli myös huomattavasti suorituskykyisempi kuin rakennettu AR-laite, joten testiohjelman käyttö oli helpompaa ja testien suorittamiseen ja toistamiseen käytettävä aika oli merkittävästi lyhyempi.

Testi 4, jossa mitattiin pelialueen tunnistavan algoritmin suoritusaikaa, suoritettiin käyttämällä esiteltyä AR-laitetta. Tämä poikkeus oli tehtävä tämän testin kohdalla,

jotta saataisiin luotettavia tuloksia suoritusajoista itse kiinnostuksen kohteena olevalle laitteelle.

Testeissä käytettiin testejä varten erikseen luotua testiohjelmaa. Testiohjelma lukee testikuvat ennalta määritellyistä kansioista, jonka jälkeen se syöttää kuvat oikeassa muodossa algoritmille. Testiohjelma tallentaa pelialueen tunnistavan algoritmin antamat tulokset ja suoritusajat. Näistä tuloksista ohjelma laskee myös lisätietoja tuloksien ymmärtämisen helpottamiseksi, kuten esimerkiksi kuvien tunnistuksen luokittelun onnistuneisiin ja epäonnistuneisiin ja keskimääräiset virheet kulmien paikoissa. Testiohjelma muodostaa suorituksen päätyttyä erillisen raportin, josta käyvät ilmi kaikki testiohjelman keräämät ja laskemat tulokset.

5.2. Tulokset

Tuloksissa raportoidaan jokaisesta testistä kuvakohtaisesti löydetyt kulmat, löydetyt virheelliset kulmat ja virheellisten kulmien virheen keskiarvo. Oikeaksi tunnistamiseksi luokittelun suurimmaksi sallituksi virhe-etäisyydeksi on asetettu arvo 6. Raportoinnissa käytetyt lyhenteet löytyvät taulukosta 2.

Taulukko 2. Käytettyjen lyhenteiden määrittelyt.

Lyhenne	**Selitys**
OTK	Oikein tunnistetut kulmat
VTK	Väärin tunnistetut kulmat
VTK_VKA	VTK:n virheen keskiarvo
SA_KA	Suoritusajan keskiarvo

5.2.1. Testi 1 – Tarkkuus hyvissä olosuhteissa

Saatujen tuloksien mukaan algoritmi toimii luotettavasti molemmilla asetuksilla hyvissä olosuhteissa. Kuvassa 7 on tapahtunut yksi kulman virheellinen paikantaminen, mutta virhe-etäisyyden ollessa matala tämä virhe ei olisi vaarantanut ohjelman toimintaa vakavasti.

Taulukko 3. Testin 1 tulokset.

Kuva	**Asetukset 1 (Canny)**			**Asetukset 2 (Otsu)**		
	OTK	**VTK**	**VTK_VKA**	**OTK**	**VTK**	**VTK_VKA**
1	4	0	-	4	0	-
2	4	0	-	4	0	-
3	4	0	-	4	0	-
4	4	0	-	4	0	-
5	4	0	-	4	0	-
6	4	0	-	4	0	-
7	3	1	7,81	4	0	-
8	4	0	-	4	0	-

5.2.2. Testi 2 – Tarkkuus kulma peitettynä

Kun pelialueen yksi kulma asetetaan peittoon ja peitettyä aluetta kasvatetaan aina seuraavassa kuvassa, on tuloksissa nähtävissä eroja eri algoritmin asetusten välillä. Asetus 1 antaa moitteettomia tuloksia 4 ensimmäisellä kuvalla. Kuvissa 5-7 tapahtuu yhden kulman tunnistuksen virhe, mutta virhe-etäisyys on niin pieni, että virhe ei olisi vaarantanut ohjelman käyttöä. Kuvassa 8 tunnistuksen tarkkuus romahtaa täysin. Asetuksella 2 kulmat tunnistetaan oikein kuvissa 1-3, mutta sen jälkeen tarkkuus romahtaa niin kriittisesti, että ohjelman käyttö olisi hyvin vaikeaa tai mahdotonta.

Taulukko 4. Testin 2 tulokset.

Kuva	**Asetukset 1 (Canny)**			**Asetukset 2 (Otsu)**		
	OTK	**VTK**	**VTK_VKA**	**OTK**	**VTK**	**VTK_VKA**
1	4	0	-	4	0	-
2	4	0	-	4	0	-
3	4	0	-	4	0	-
4	4	0	-	2	2	37.76
5	3	1	9,22	3	1	42,52
6	3	1	11,70	2	2	39,84
7	3	1	15,52	1	3	33,95
8	1	2	73,17	1	3	32,87

5.2.3. Testi 3 – Tarkkuus hämärissä olosuhteissa

Hämärissä olosuhteissa kuvattu testikuvasarja tuottaa selviä eroja eri asetuksien tuottamien tulosten välille. Asetuksilla 1 algoritmi epäonnistuu täydellisesti matalan valaistuksen kuvissa 1-5, mutta toimii oikein kuvissa 6-8. Asetukset 2 toimivat kaikissa kuvissa paitsi ensimmäisessä, mutta siinäkin kulman tunnistuksen virhe on niin pieni, että se ei haittaa ohjelman toimintaa.

Taulukko 5. Testin 3 tulokset

Kuva	**Asetukset 1 (Canny)**			**Asetukset 2 (Otsu)**		
	OTK	**VTK**	**VTK_VKA**	**OTK**	**VTK**	**VTK_VKA**
1	0	0	-	3	1	8,00
2	0	0	-	4	0	-
3	0	0	-	4	0	-
4	0	0	-	4	0	-
5	0	0	-	4	0	-
6	4	0	-	4	0	-
7	4	0	-	4	0	-
8	4	0	-	4	0	-

5.2.4. Testi 4 – Suoritusaika eri resoluution kuvilla

Tulokset näyttävät odotetusti parempaa suoritusaikaa pienemmän resoluution kuville. Suoritusaika vaikuttaa olevan lähes suoraan verrannollinen pikselimäärään näissä havainnoissa. Testin tulokset näyttävät nopeampia suoritusaikoja asetukselle 2, mikä oli odotettua, koska kynnystäminen on yksinkertaisempi operaatio kuin Canny-reunantunnistusalgoritmin avulla segmentointi. Näissä tuloksissa asetuksen 1 suoritusaika on 1,53–1,56-kertainen verrattuna asetuksen 2 suoritusaikaan.

Taulukko 6. Testin 4 tulokset.

Käytetty materiaali	**Asetukset 1 (Canny)**	**Asetukset 2 (Otsu)**
	SA_KA	**SA_KA**
Testimateriaali 1(320x240px)	0,0424 s	0,0273 s
Testimateriaali 1(640x480px)	0,1719 s	0,1122 s
Testimateriaali 1(1280x960px)	0,7086 s	0,4628 s

5.2.5. Testi 5 – Tarkkuus reuna peitettynä

Tulokset näyttävät parempaa sietokykyä asetukselle 1 tilanteessa, jossa pelialueen reunan yli työntyy häiritsevä kappale. Asetus 1 toimii kaikissa muissa kuvissa paitsi kahdessa viimeisessä kuvassa, joista ensimmäisessä este on läpäissyt pelialueen lähes kokonaan ja toisessa pelialue on kokonaan leikkaantunut kahteen osaan esteen läpäistyä sen. Asetus 2 menettää toimintakykynsä jo kuvan 4 kohdalla ja tuottaa sen jälkeen vahvasti virheellisiä tuloksia.

Taulukko 7. Testin 5 tulokset.

Kuva	**Asetukset 1 (Canny)**			**Asetukset 2 (Otsu)**		
	OTK	**VTK**	**VTK_VKA**	**OTK**	**VTK**	**VTK_VKA**
1	4	0	-	4	0	-
2	4	0	-	4	0	-
3	4	0	-	4	0	-
4	4	0	-	3	1	66,03
5	4	0	-	2	2	38.69
6	4	0	-	2	2	48.24
7	2	2	93,01	2	2	55.15
8	2	2	98,51	2	2	57,26

5.2.6. Testi 6 – Tarkkuus pelialue kallistettuna

Kallistettaessa pelialuetta asteittain saadaan tuloksiin eri asetusten välillä syntymään huomattavia eroja. Asetus 1 toimii oikein kuvissa 1-4 ja siedettävillä virheillä kuvissa 5-7. Asetuksen 1 tuottama tulos on selkeästi virheellinen vasta erittäin voimakkaasti kallistetussa kuvassa 8. Asetus 2 tuottaa kuvissa 3-8 vahvasti

virheellisiä tuloksia. Asetus 2 on tällä testimateriaalilla selkeästi heikompi kallistetun pelialueen tunnistamisessa.

Taulukko 8. Testin 6 tulokset.

Kuva	**Asetukset 1 (Canny)**			**Asetukset 2 (Otsu)**		
	OTK	**VTK**	**VTK_VKA**	**OTK**	**VTK**	**VTK_VKA**
1	4	0	-	4	0	-
2	4	0	-	4	0	-
3	4	0	-	3	1	55,46
4	4	0	-	1	3	61,67
5	2	2	6,36	2	2	70,03
6	2	2	8,56	1	3	69,37
7	2	2	12,43	0	4	67,43
8	0	3	31,46	0	4	57,26

6. POHDINTA

Vaikka testeissä käytettyjen testimateriaalien suuruus ja monipuolisuus eivät riitä varmojen tai tilastollisesti merkittävien päätelmien esittämiseen algoritmin suorituskyvystä, niin testien tulokset kuitenkin mahdollistavat karkean arvioinnin algoritmin suorituskyvystä eri asetuksien ollessa käytössä. Joka tapauksessa testeistä saadut tulokset antavat uusia suuntia ohjelman jatkokehitykselle.

6.1. Testien tulokset

Testeissä käytetyt testimateriaalit eivät olleet tarpeeksi suuria ja monipuolisia jotta olisi saavutettu tilastollisesti merkittäviä tuloksia tai varmoja tietoja eri asetusten suorituskyvystä eri tilanteissa. Vaikka testeissä havaitut eroavaisuudet eri algoritmin asetusten välillä olivat paikoin huomattavia, niin edellä mainitun seikan vuoksi kaikkia tässä osiossa esiteltäviä päätelmiä on kohdeltava ainoastaan suuntaa antavina.

Algoritmi havaittiin odotusten mukaisesti hyvin toimivaksi testissä 1, jossa olosuhteet olivat hyviä. Testissä 1 havaittiin asetuksella 1 yhdessä kuvassa yhden kulman vähäinen virheellinen tunnistus, mutta se ei pienuudestaan johtuen olisi haitannut ohjelman käyttöä. Testikuvien vähyyden vuoksi on myös mahdotonta sanoa, että onko tämä virhe toistuva vai satunnainen tapahtuma.

Testissä 2, jossa pelialueen yhtä kulmaa asetettiin peittoon, toimi algoritmi eri asetuksilla huomattavasti eri tavalla. Asetus 2 toimi vielä oikein, kun pelialueesta oli peitossa 7%, mutta tuloksien laatu tippui rajusti, kun pelialueesta oli peitossa 16% tai enemmän. Asetus 1 toimi oikein pelialueen peiton ollessa 16% tai vähemmän. Pelialueen peiton kasvaessa arvoihin 31%, 49% ja 62% huononi algoritmin tuottama tulos, mutta vain yhden kulman kohdalta ja silloinkaan ei virhe ollut vakava. Täydellinen tunnistamiskyvyn romahdus tapahtui asetuksella 1 pelialueen peiton ollessa 78%. Näiden tulosten perusteella voidaan esittää päätelmä, että asetus 1 toimii siedettävästi peitettäessä yksi kulma ja enintään 62% pelialueesta ja asetus 2 toimii siedettävästi peitettäessä yksi kulma ja enintään 7% pelialueesta.

Valaistusolosuhteiden ollessa huonot, mitä testattiin testissä 3, tuotti algoritmin asetus 2 parempia tuloksia. Asetuksella 2 algoritmi tuotti oikeita tuloksia kaikissa kuvissa paitsi kaikkein huonoiten valaistussa, mutta siinäkin havaittu virhe oli melko pieni. Asetuksella 1 algoritmi tunnisti kuvasta pelialueen vasta kuvassa 6 ja sitä seuraavissa paremman valaistuksen kuvissa. Tutkimalla testimateriaalin 3 kuvia 2 ja 6 saatiin niille pelialueen ja taustan välillä olevaksi harmaasävyjen kirkkauden keskiarvon eroksi arvot 10 ja 42. Mahdolliset kirkkauden arvot olivat tässä tapauksessa 0-255. Tämä johtaa seuraavaan päätelmään: Asetuksella 2 algoritmi vaatii pelialueen luotettavaan tunnistamiseen pelialueen harmaasävyn olevan 10 yksikköä kirkkaampi, kun asetuksella 1 sama vaatimus on vähintään 42 yksikköä.

Testin 4 tulokset osoittavat asetuksen 1 olevan hitaampi kuin asetuksen 2. Tämä oli odotettua, koska Canny-reunantunnistusalgoritmin soveltaminen oli monimutkaisempi operaatio kuin yksinkertainen kuvan kynnystäminen.

Pelialueen tunnistavan algoritmin suorittaminen on vain yksi ohjelman toiminnoista, joka on tehtävä ennen kuin lopullinen informaatio saadaan näkymään käyttäjälle. Suoritusaikaa kuluu paljon muihin toimintoihin, kuten kuvan saamiseen laitteen kameralta, kuvan piirtämiseen laitteen näytölle ja pallon liikkumisen

laskemiseen. Vaikka näitä seikkoja ei otettu huomioon sovelluksen testaamisessa, on testin 4 tuottamien tulosten perusteella on kuitenkin mahdollista tehdä karkeita päätelmiä luodun sovelluksen ja esitellyn AR-laitteen rajoituksista. Esimerkiksi saaduista tuloksista voidaan päätellä, että käyttämällä asetuksia 1 ja resoluutiota 320x240 on kuvataajuuden 30 kuvaa sekunnissa saavuttaminen mahdotonta, koska pelkän pelialueen tunnistaminen kuvasta vie enemmän aikaa kuin tässä tapauksessa olisi yhden kuvan koko prosessoinnille varattu.

Testissä 5 havaittiin algoritmin toimivan asetuksella 1 luotettavammin kuin asetuksella 2. Käytettäessä asetusta 1 tuottaa algoritmi virheellisiä tuloksia kuvasarjan kahdessa viimeisessä kuvassa, joissa reunan yli työntynyt este on lähes läpäissyt tai kokonaan läpäissyt pelialueen. Tämä käyttäytyminen oli odotettua, koska algoritmi ohjelmoitiin etsimään suurinta yhtenäistä pelialuetta. Asetuksella 2 algoritmi tuottaa vahvasti virheellisiä tuloksia kuvassa 4 ja sen jälkeen. Kuvassa 4 pelialueen peittoprosentti on 20%, mikä on samansuuntainen peiton suuruus, joka aiheuttaa asetuksen 2 tuottamat huonot tulokset testissä 2.

Testissä 6 havaittiin algoritmin kestävän eri lailla pelialueen kallistamista eri asetuksilla. Asetuksella 1 algoritmi kestää hyvin kallistusta ennen kuin se tuottaa tuottamaan ensin kevyesti virheellisiä ja sitten raskaasti virheellisiä tuloksia. Asetus 2 tuottaa raskaasti virheellisiä tuloksia jo pienillä kallistuksen määrillä. Tämän testimateriaalin tuloksien perusteella on ilmeistä todeta, että asetus 1 toimii huomattavasti paremmin, jos pelialue ei ole kohtisuorassa kulmassa kameraan nähden.

Kokonaisuutena testeissä yleisenä linjana oli asetuksen 1 paremmuus suurimmassa osassa testejä. Asetus 2 oli parempi ainoastaan huonoissa valaistusolosuhteissa ja lyhyemmässä suoritusajassa.

6.2. Tulevaisuuden työ

Tulevaisuudessa ohjelmaa voisi testata tarkemmin. Koska ohjelman toiminnasta testattiin melko tarkkaan rajattua osaa, niin mahdollisuuksia uusille testeille on monia.

Tärkeitä testauskohteita olisi esimerkiksi tarkempien algoritmin toiminnan rajojen selvittäminen valaistukselle, reunan ja kulman peittämiselle ja kallistukselle. Lisäksi tärkeää olisi selvittää, mistä testissä 1 ja testissä 3 havaitut yksittäiset pienet virheet johtuvat. Algoritmia ei testattu yhdessäkään testissä sillä oletuksella, että pelialuetta ei olisi löydetty edellisestä kuvasta. Algoritmin tulosten mittaaminen käyttämällä tätä alkuoletusta olisi kokonaan uusi polku hyödynnettäväksi testaamiseen.

Ohjelman käytön ja testauksen yhteydessä ilmeni siinä lukuisia puutteita, mikä avaa uusia mahdollisuuksia jatkokehitykselle.

Toteutettuun sovellukseen pystyy helposti lisäämään uusia segmentointimenetelmiä ja suodattimia. Ohjelmaa voisi myös kehittää siihen suuntaan, että se kykenisi itse valitsemaan olosuhteiden mukaan parhaimman suodattimen ja segmentoinnin.

Rakennetussa AR-laitteessa on neliytiminen prosessori. Tästä kuitenkin hyödynnetään vain yhtä ydintä. Tulevaisuudessa voisi ohjelman kehittää käyttämään rinnakkaislaskentaa hyödykseen, jotta olisi mahdollista käyttää ohjelmassa suurempaa resoluutiota tai nostaa ohjelman käyttämää kuvataajuutta.

Pelialueen tunnistavaa algoritmia voidaan jatkokehittää monella tapaa: Kehittyneempi kulmien karsinta, hienostuneempi peitettyjen kulmien paikan

arviointi ja kahtia jakautuneen pelialueen tunnistaminen yhdeksi ovat tärkeimpien parannettavien kohteiden listalla.

7. YHTEENVETO

Lisätty todellisuus on noussut laitteistojen kehittymisen ja halpenemisen myötä kiinnostavaksi kohteeksi myös alaa tutkivan tiedeyhteisön ulkopuolella. Tälle työlle motiivin antoi se, että lisätyn todellisuuden sovellukset ovat yleistyneet arkipäivän elämässä, ja on ensiarvoisen tärkeää tutkia ja kehittää uusia tapoja, joilla ne kykenevät ympäröivän maailman havainnoinnin lisäksi prosessoimaan havaitsemiaan tietoja.

Tässä työssä esiteltiin useita lisätyn todellisuuden laitteistoja ja lisätyn todellisuuden sovelluksia, joita on aikaisemmin kehitetty sotilaskäyttöön, viihdetarkoituksiin, opetuskäyttöön ja lääketieteellisiin tarkoituksiin. Työssä perehdyttiin myös lukuisiin kuvan matalan tason prosessoinnin menetelmiin, jotka toimivat pohjana monissa kuvankäsittelyä vaativissa sovelluksissa ja kuvan korkean tason prosessoinnin menetelmissä.

Tämä työ esitteli luodun lisätyn todellisuuden pelisovelluksen ja AR-laitteen, jossa sovellusta on mahdollista käyttää. Luodussa pelisovelluksessa ohjelma tunnistaa pelialueen ja lisää laitteen näytölle pelialueeseen virtuaalisen pallon, jota sovelluksen käyttäjän on kosketettava kädellään saadakseen pisteitä.

Luodun sovelluksen pelialueen tunnistavaa algoritmia testattiin kuudella erilaisella testillä, joissa mitattiin algoritmin toimintaa kahta erilaista segmentointiasetusta käyttämällä. Testeistä saatujen tuloksien perusteella voitiin suuntaa antavasti todeta Canny-reunantunnistusalgoritmia hyödyntäneen segmentointiratkaisun olleen parempi tilanteissa, joissa pelialue oli peitetty tai kallistunut, kun taas Otsun menetelmän käyttö tuotti parempi tuloksia hämärissä olosuhteissa sekä nopeamman suoritusajan. Testien tuloksien perusteella havaittiin lukuisia mahdollisuuksia ohjelman jatkokehitykselle, kuten rinnakkaislaskennan hyödyntäminen, kehittyneempien segmentointimenetelmien käyttö ja älykkäämpi kulmien karsinta.

8. LÄHTEET

[1] Nee A, Ong S, Chryssolouris G & Mourtzis D (2012) Augmented reality applications in design and manufacturing. CIRP Annals-Manufacturing Technology

[2] Zollmann S, Hoppe C, Kluckner S, Poglitsch C, Bischof H & Reitmayr G (2014) Augmented reality for construction site monitoring and documentation. Proc IEEE 102(2): 137-154.

[3] Lee K (2012) Augmented reality in education and training. TechTrends 56(2): 13-21.

[4] Livingston MA, Rosenblum LJ, Julier SJ, Brown D, Baillot Y, Swan I, Gabbard JL & Hix D (2002) An augmented reality system for military operations in urban terrain.

[5] Hughes CE, Stapleton CB, Hughes DE & Smith EM (2005) Mixed reality in education, entertainment, and training. IEEE Comput Graphics Appl 25(6): 24-30.

[6] Van Krevelen D & Poelman R (2010) A survey of augmented reality technologies, applications and limitations. International Journal of Virtual Reality 9(2): 1.

[7] Cakmakci O & Rolland J (2006) Head-worn displays: a review. Journal of display technology 2(3): 199-216

[8] Barfield W (2015) Fundamentals of wearable computers and augmented reality. , CRC Press Henderson S & Feiner S (2011) Exploring the benefits of augmented reality documentation for maintenance and repair. IEEE Trans Visual Comput Graphics 17(10): 1355-1368.

[9] Chellappan KV, Erden E & Urey H (2010) Laser-based displays: a review. Appl Opt 49(25): F79-F98.

[10] Bayer MM (2002) Retinal scanning display: a novel HMD approach for army aviation. AeroSense 2002. International Society for Optics and Photonics: 202-213.

[11] Kim S, Prinzel LJ, Kaber DB, Alexander AL, Stelzer EM, Kaufmann K & Veil T (2011) Multidimensional measure of display clutter and pilot performance for advanced head-up display. Aviat Space Environ Med 82(11): 1013-1022.

[12] Henderson SJ & Feiner S (2009) Evaluating the benefits of augmented reality for task localization in maintenance of an armored personnel carrier turret. Mixed and Augmented Reality, 2009. ISMAR 2009. 8th IEEE International Symposium on. IEEE: 135-144.

[13] Webel S, Bockholt U, Engelke T, Gavish N, Olbrich M & Preusche C (2013) An augmented reality training platform for assembly and maintenance skills. Robotics and Autonomous Systems 61(4): 398-403.

[14] https://www.microsoft.com/microsoft-hololens/en-us Noudettu 5.1.2017.

[15] Dey A & Sandor C (2014) Lessons learned: Evaluating visualizations for occluded objects in handheld augmented reality. International Journal of Human-Computer Studies 72(10): 704-716.

[16] Wagner D & Schmalstieg D (2006) Handheld augmented reality displays. IEEE Virtual Reality Conference (VR 2006). , IEEE: 321-321.

[17] Carmigniani J, Furht B, Anisetti M, Ceravolo P, Damiani E & Ivkovic M (2011) Augmented reality technologies, systems and applications. Multimedia Tools Appl 51(1): 341-377

[18] Bimber O & Raskar R (2005) Spatial augmented reality: merging real and virtual worlds. , CRC press

[19] Lv Z, Halawani A, Feng S, Ur Réhman S & Li H (2015) Touch-less interactive augmented reality game on vision-based wearable device. Personal and Ubiquitous Computing 19(3-4): 551-567

[20] Hughes CE, Stapleton CB, Hughes DE & Smith EM (2005) Mixed reality in education, entertainment, and training. IEEE Comput Graphics Appl 25(6): 24-30

[21] Nilsen T, Linton S & Looser J (2004) Motivations for augmented reality gaming. Proceedings of FUSE 4: 86-93.

[22] Armstrong S & Morrand K (2016) Ghost Hunter–An Augmented Reality Ghost Busting Game. International Conference on Virtual, Augmented and Mixed Reality. , Springer: 671-678.

[23] Botella C, Breton-Lopez J, Quero S, Baños RM, Garcia-Palacios A, Zaragoza I & Alcaniz M (2011) Treating cockroach phobia using a serious game on a mobile phone and augmented reality exposure: A single case study. Comput Hum Behav 27(1): 217-227.

[24] Specht M, Ternier S & Greller W (2011) Mobile augmented reality for learning: A case study. Journal of the Research Center for Educational Technology 7(1): 117-127.

[25] Ferdinand P, Müller S, Ritschel T & Wechselberger U (2005) The Eduventure–A new approach of digital game based learning combining virtual and mobile augmented reality games episodes. Pre-Conference Workshop “Game based Learning” of DeLFI 2005 and GMW 2005 Conference, Rostock. 13.

[26] Lee K (2012) Augmented reality in education and training. TechTrends 56(2): 13-21.

[27] Yuen S, Yaoyuneyong G & Johnson E (2011) Augmented reality: An overview and five directions for AR in education. Journal of Educational Technology Development and Exchange 4(1): 119-140.

[28] Agogi E (2011) Augmented Reality in Education. Proceedings of Science Center to Go Workshop, Athens, Greece: EDEN-2011 Open Classroom Conference. : 1-88.

[29] http://www.sctg.eu/ Noudettu 31.1.2017

[30] https://www.youtube.com/watch?v=4JTCabfxEcw Noudettu 31.1.2017

[31] Piekarski W & Thomas B (2002) ARQuake: the outdoor augmented reality gaming system. Commun ACM 45(1): 36-38.

[32] Ma M, Jain LC & Anderson P (2014) Future trends of virtual, augmented reality, and games for health. In: Anonymous Virtual, Augmented Reality and Serious Games for Healthcare 1. , Springer: 1-6.

[33] Yohan SJ, Julier S, Baillot Y, Lanzagorta M, Brown D & Rosenblum L (2000) Bars: Battlefield augmented reality system. In NATO Symposium on Information Processing Techniques for Military Systems. , Citeseer.

[34] Henderson SJ & Feiner S (2009) Evaluating the benefits of augmented reality for task localization in maintenance of an armored personnel carrier turret. Mixed and Augmented Reality, 2009. ISMAR 2009. 8th IEEE International Symposium on. , IEEE: 135-144.

[35] Henderson S & Feiner S (2011) Exploring the benefits of augmented reality documentation for maintenance and repair. IEEE Trans Visual Comput Graphics 17(10): 1355-1368.

[36] Webel S, Bockholt U, Engelke T, Gavish N, Olbrich M & Preusche C (2013) An augmented reality training platform for assembly and maintenance skills. Robotics and Autonomous Systems 61(4): 398-403.

[37] LaViola Jr JJ, Williamson BM, Brooks C, Veazanchin S, Sottilare R & Garrity P (2015) Using Augmented Reality to Tutor Military Tasks in the Wild. Interservice/Industry Training Simulation and Education Conference.

[38] Julier S, Lanzagorta M, Baillot Y, Rosenblum L, Feiner S, Hollerer T & Sestito S (2000) Information filtering for mobile augmented reality. Augmented Reality, 2000.(ISAR 2000). Proceedings. IEEE and ACM International Symposium on. , IEEE: 3-11.

[39] Wang X, Truijens M, Hou L, Wang Y & Zhou Y (2014) Integrating Augmented Reality with Building Information Modeling: Onsite

construction process controlling for liquefied natural gas industry. Autom Constr 40: 96-105.

[40] Yuvana P & NVK Ramesh N (2015) Developing Killer Apps for Industrial Augmented Reality. Global Journal of Research In Engineering 14(7).

[41] Pal NR & Pal SK (1993) A review on image segmentation techniques. Pattern Recognit 26(9): 1277-1294.

[42] Sezgin M (2004) Survey over image thresholding techniques and quantitative performance evaluation. Journal of Electronic imaging 13(1): 146-168.

[43] Rosenfeld A & De La Torre P (1983) Histogram concavity analysis as an aid in threshold selection. IEEE Trans Syst Man Cybern (2): 231-235.

[44] dos Anjos A & Shahbazkia H (2008) Bi-Level Image Thresholding-A Fast Method. BIOSIGNALS (2). : 70-76.

[45] Ridler T & Calvard S (1978) Picture thresholding using an iterative selection method. IEEE Trans Syst Man Cybern 8(8): 630-632.

[46] Otsu N (1975) A threshold selection method from gray-level histograms. Automatica 11(285-296): 23-27.

[47] Kittler J & Illingworth J (1986) Minimum error thresholding. Pattern Recognit 19(1): 41-47.

[48] Jawahar C, Biswas PK & Ray A (1997) Investigations on fuzzy thresholding based on fuzzy clustering. Pattern Recognit 30(10): 1605-1613.

[49] Kapur JN, Sahoo PK & Wong AK (1985) A new method for gray-level picture thresholding using the entropy of the histogram. Computer vision, graphics, and image processing 29(3): 273-285.

[50] Li CH & Lee C (1993) Minimum cross entropy thresholding. Pattern Recognit 26(4): 617-625.

[51] Shanbhag AG (1994) Utilization of information measure as a means of image thresholding. CVGIP: Graphical Models and Image Processing 56(5): 414-419.

[52] Hertz L & Schafer RW (1988) Multilevel thresholding using edge matching. Computer Vision, Graphics, and Image Processing 44(3): 279-295.

[53] Tsai W (1985) Moment-preserving thresolding: A new approach. Computer Vision, Graphics, and Image Processing 29(3): 377-393.

[54] Chanda B & Majumder DD (1988) A note on the use of the graylevel co-occurrence matrix in threshold selection. Signal Process 15(2): 149-167.

[55] Sauvola J & Pietikäinen M (2000) Adaptive document image binarization. Pattern Recognit 33(2): 225-236.

[56] White JM & Rohrer GD (1983) Image thresholding for optical character recognition and other applications requiring character image extraction. IBM Journal of research and development 27(4): 400-411.

[57] Lloyd S (1982) Least squares quantization in PCM. IEEE Trans Inf Theory 28(2): 129-137.

[58] Hartigan JA & Wong MA (1979) Algorithm AS 136: A k-means clustering algorithm. Journal of the Royal Statistical Society.Series C (Applied Statistics) 28(1): 100-108.

[59] Ray S & Turi RH (1999) Determination of number of clusters in k-means clustering and application in colour image segmentation. Proceedings of the 4th international conference on advances in pattern recognition and digital techniques. , Calcutta, India: 137-143.

[60] Pappas TN (1992) An adaptive clustering algorithm for image segmentation. IEEE Transactions on signal processing 40(4): 901-914.

[61] Chen CW, Luo J & Parker KJ (1998) Image segmentation via adaptive K-mean clustering and knowledge-based morphological operations with biomedical applications. IEEE

[62] Ng H, Ong S, Foong K, Goh P & Nowinski W (2006) Medical image segmentation using k-means clustering and improved watershed algorithm. 2006 IEEE Southwest Symposium on Image Analysis and Interpretation. , IEEE: 61-65.

[63] Patel BC & Sinha G (2010) An adaptive k-means clustering algorithm for breast image segmentation. International Journal of Computer Applications 10(4): 35-38.Canny J (1986) A computational approach to edge detection. IEEE Trans Pattern Anal Mach Intell (6): 679-698.

[64] Dunn JC (1973) A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters.

[65] Chuang K, Tzeng H, Chen S, Wu J & Chen T (2006) Fuzzy c-means clustering with spatial information for image segmentation. Comput Med Imaging Graphics 30(1): 9-15.

[66] Liew AW & Yan H (2003) An adaptive spatial fuzzy clustering algorithm for 3-D MR image segmentation. IEEE Trans Med Imaging 22(9): 1063-1075.

[67] Li BN, Chui CK, Chang S & Ong SH (2011) Integrating spatial fuzzy clustering with level set methods for automated medical image segmentation. Comput Biol Med 41(1): 1-10.

[68] Cai W, Chen S & Zhang D (2007) Fast and robust fuzzy c-means clustering algorithms incorporating local information for image segmentation. Pattern Recognit 40(3): 825-838.

[69] Mobahi H, Rao SR, Yang AY, Sastry SS & Ma Y (2011) Segmentation of natural images by texture and boundary compression. International journal of computer vision 95(1): 86-98.

[70] Yang AY, Wright J, Ma Y & Sastry SS (2008) Unsupervised segmentation of natural images via lossy data compression. Comput Vision Image Understanding 110(2): 212-225

[71] Adams R & Bischof L (1994) Seeded region growing. IEEE Trans Pattern Anal Mach Intell 16(6): 641-647.

[72] Lin Z, Jin J & Talbot H (2000) Unseeded region growing for 3D image segmentation. Selected papers from the Pan-Sydney workshop on Visualisation-Volume 2. , Australian Computer Society, Inc.: 31-37.

[73] Beucher S (1992) The watershed transformation applied to image segmentation. SCANNING MICROSCOPY-SUPPLEMENT- : 299-299.

[74] Meyer F & Beucher S (1990) Morphological segmentation. Journal of visual communication and image representation 1(1): 21-46.

[75] Nock R & Nielsen F (2004) Statistical region merging. IEEE Trans Pattern Anal Mach Intell 26(11): 1452-1458.

[76] Li H, Gu H, Han Y & Yang J (2009) An efficient multiscale SRMMHR (Statistical Region Merging and Minimum Heterogeneity Rule) segmentation method for high-resolution remote sensing imagery. IEEE Journal of Selected Topics in Applied earth observations and remote sensing 2(2): 67-73.

[77] Anil P & Natarajan S (2010) Automatic road extraction from high resolution imagery based on statistical region merging and skeletonization. International Journal of Engineering Science and Technology 2(3): 165-171.

[78] Peng B, Zhang L & Zhang D (2013) A survey of graph theoretical approaches to image segmentation. Pattern Recognit 46(3): 1020-1038.

[79] Shi J & Malik J (2000) Normalized cuts and image segmentation. IEEE Trans Pattern Anal Mach Intell 22(8): 888-905.

[80] Felzenszwalb PF & Huttenlocher DP (2004) Efficient graph-based image segmentation. International journal of computer vision 59(2): 167-181.

[81] Zahn CT (1971) Graph-theoretical methods for detecting and describing gestalt clusters. IEEE Trans Comput 100(1): 68-86.

[82] Grady L (2006) Random walks for image segmentation. IEEE Trans Pattern Anal Mach Intell 28(11): 1768-1783.

[83] Canny J (1986) A computational approach to edge detection. IEEE Trans Pattern Anal Mach Intell (6): 679-698.

[84] Maini R & Aggarwal H (2009) Study and comparison of various image edge detection techniques. International journal of image processing (IJIP) 3(1): 1-11.

[85] Shrivakshan G & Chandrasekar C (2012) A comparison of various edge detection techniques used in image processing. IJCSI International Journal of Computer Science Issues 9(5): 272-276.

[86] Deriche R (1987) Using Canny's criteria to derive a recursively implemented optimal edge detector. International journal of computer vision 1(2): 167-187.

[87] Sobel I (1990) An isotropic 3× 3 image gradient operator. Machine Vision for three-demensional Sciences.

[88] Gonzalez RC & Woods RE (2007) Image processing. Digital image processing 2.

[89] Roberts LG (1963) Machine perception of three-dimensional solids.

[90] Duda RO & Hart PE (1972) Use of the Hough transformation to detect lines and curves in pictures. Commun ACM 15(1): 11-15.

[91] Ballard DH (1981) Generalizing the Hough transform to detect arbitrary shapes. Pattern Recognit 13(2): 111-122.

[92] Kiryati N, Eldar Y & Bruckstein AM (1991) A probabilistic Hough transform. Pattern Recognit 24(4): 303-316.

[93] Harris C & Stephens M (1988) A combined corner and edge detector. Alvey vision conference. Citeseer 15: 10.5244.

[94] Smith SM & Brady JM (1997) SUSAN—A new approach to low level image processing. International journal of computer vision 23(1): 45-78.

[95] https://www.raspberrypi.org/documentation/hardware/ Noudettu 31.1.2017

[96] http://www.numpy.org/ Noudettu 31.1.2017

[97] http://opencv.org/ Noudettu 31.1.2017

[98] Satoshi Suzuki and others. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30(1):32–46, 1985.